**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр КОВАЛЬ
«___»_____2020 р.

# Дипломна робота

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»**
**спеціальності 122 «Комп'ютерні науки та інформаційні технології»**
**на тему: «**Інструментальні засоби транспортної телематики**»**

Виконав:
студент IV курсу, групи ТМ-61
Пащенко Дмитро Олександрович _____
Керівник:
доцент, кандидат економічних наук
Гусєва Ірина Ігорівна _____
Рецензент:
доцент, кандидат технічних наук
Веремійчук Юрій Андрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

**Національний технічний університет України**
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки 122 Комп’ютерні науки та інформаційні технології
Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
(підпис)

”___”_____2020р.

# ЗАВДАННЯ
**на дипломну роботу студенту**

_____Пащенко Дмитру Олександровичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи Інструментальні засоби транспортної телематики_____
керівник роботи_ Гусєва Ірина Ігорівна, доцент, кандидат економічних наук__
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _12.06.20_____

3. Вихідні дані до роботи _Swift, JavaScript, SQL, IOS_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Проаналізувати область та методи використання використання інструментальних засобів транспортної телематики. Розробити програмну систему, що складається з бази даних, мобільного додатку для зчитування показників сенсорів мобільного пристрою та веб-додаток аналізу та адміністрування отриманих даних. Провести комплексне тестування системи. Проаналізувати його результати, зробити висновки та провести налагодження системи._____

5. Перелік ілюстративного матеріалу
__“Інструментальні засоби транспортної телематики”, “Постановка задачі”, “Основні сенсори смартфону”, “Використання отриманої інформації”, “Основні проблеми при проектуванні системи - 1”, “Основні проблеми при проектуванні системи - 2”, “ Основні проблеми при проектуванні системи - 3”, “ Виявлення аномалій дорожнього покриття ”, “Засоби розробки”, “Як працює система?”, “Як працює система?”, “74%”, “Мобільний додаток користувача”, “Мобільний додаток адміністратора”, “Веб-додаток адміністратора - 1”, “Веб-додаток адміністратора - 2”, “Веб-додаток адміністратора - 3”, “Висновки”__

6. Дата видачі завдання ”_11_”___жовтня___2020 р.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітки |
|---|---|---|---|
| 1. | Затвердження теми роботи | 03.02.20 | |
| 2. | Вивчення та аналіз задачі | 13-19.04.20 | |
| 3. | Розробка архітектури та загальної структури системи | 20-26.04.20 | |
| 4. | Розробка структур окремих підсистем | 27.04-03.05.20 | |
| 5. | Програмна реалізація системи | 04-13.05.20 | |
| 6. | Оформлення пояснювальної записки | 14-16.05.20 | |
| 7. | Захист програмного продукту | 17.05.20 | |
| 8. | Передзахист | 12.06.20 | |
| 9. | Захист | 15.06.20 | |

Студент _____ _____Дмитро ПАЩЕНКО_____
(підпис) (ім’я та прізвище)

Керівник роботи _____ _____Ірина ГУСЄВА_____
(підпис) (ім’я та прізвище)

# АНОТАЦІЯ

Метою цієї роботи є створення системи зчитування даних з сенсорів смартфону та аналізу інформації для подальшого її використання у сфері транспортної телематики.

Система складається з мобільного додатку, за допомогою якого зчитуються показники сенсорів, веб-додатку, для детального аналізу отриманої інформації та веб-серверу. Мобільний додаток реалізовано засобами мови Swift для операційної системи IOS. Веб-додаток створено за допомогою скриптової мови JavaScript. Веб-сервер реалізовано з використанням мов програмування Swift та SQL.

Загальний обсяг роботи: 63 сторінок, 13 таблиць, 49 ілюстрацій та 16 використаних джерел.

Ключові слова: транспортна телематика, геоінформаційна система, сенсори смартфону.

# ABSTRACT

The purpose of this work is creating a system for reading data from smartphone sensors and analysis this information for future use of transport telematics.

The system consists of a mobile application, which reads the sensor data and a web application for performing detailed analysis of the received information. The mobile application for iOS was created using the Swift programming language. The web application was created using the JavaScript programming language. The web server was implemented using Swift and SQL programming languages.

Number of pages: 63

Number of tables: 13

Number of illustrations: 49

Number of sources: 16

Key words: transport telematics, geographic information system, smartphone sensors.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ITC — інтелектуальна транспортна системи

Wi-Fi — wireless fidelity, технологія бездротової локальної мережі

GPS — global positioning system, система глобального позиціонування

СКБД — система керування базами даних

БД — база даних

ГБД — геоінформаційна база даних

CPU — central processing unit, електронний блок або інтегральна схема, яка виконує машинні інструкції, процесор

API — application programming interface, опис способів, якими одна комп'ютерна програма може взаємодіяти з іншою програмою

LLVM — low level virtual machine, проект програмної інфраструктури для створення компіляторів і супутніх їм утиліт

UML — unified modeling language, це система позначень, яку можна застосовувати для об'єктно-орієнтованого аналізу і проектування

ORM — Object-Relational Mapping, технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування

# ВСТУП

Однією з невирішених проблем сьогодні є збір та аналіз великої кількості даних. У кожній сфері існують свої методи та стратегії роботи з інформаційними потоками. Для великої кількості галузей вже було винайдено оптимальні шляхи обробки інформації й протягом тривалого часу вони не змінювалися. Але й до цього часу, гостро постає питання збору даних для дорожньо-транспортної сфери.

Область широко досліджується протягом багатьох років. Транспорт є одним з основних секторів, що значно впливає на соціально-економічне життя населення. Тому питання контролю та покращення якості транспортної системи має значну вагу. В цих умовах, транспортна телематика стала однією з найбільш перспективних і необхідних галузей. Транспортна телематика являє собою систему віддаленого контролю стану автотранспорту. На сьогоднішній день, вже існує певна кількість додатків, що працюють з транспортною телематикою. І їх кількість продовжує зростати.

Аналіз дорожньо-транспортного руху, стану доріг та поведінки водіїв на дорогах досі викликає значні труднощі, як в теорії так і на практиці. Сьогодні, автотранспортні засоби, як і дороги, оснащені сенсорами та іншими пристроями для збору інформації, до них відносяться камери спостереження для відслідковування автомобільного трафіку, різні типи радарів та відеореєстраторів. Однак, дані, що було отримано таким чином достатньо важко групувати, передавати на інші пристрої та аналізувати.

Для вирішення цього питання потрібно визначити спосіб, який допоможе швидко та максимально ефективно зібрати дані про події, що відбуваються, під час руху автотранспорту й зможе оптимальним шляхом готувати їх до аналізу. Успішним вирішенням даного питання буде створення мобільного додатку, котрий збирає дані сенсорів мобільного пристрою (акселерометра, гіроскопа, магнітометр і т.п.) та конвертує їх до вигляду, в якому їх комфортно та ефективно аналізувати.

Програмне забезпечення допоможе збирати інформацію для декількох ключових цілей:

- Аналіз стану доріг;
- Аналіз трафіку на дорогах;
- Побудова оптимальних маршрутів;
- Класифікація подій на дорозі;
- Аналіз поведінки водіїв на дорозі;

Така методика збору даних може використовуватися для всіх, без винятку, транспортних засобів, що є значною перевагою серед інших методів аналізу цієї предметної області. Отже, враховуючи мобільність даного методу збору та аналізу даних, можна зробити висновок, що побудована система є найбільш логічним варіантом вирішення задачі, що стоїть перед нами. Таким чином програмний продукт надає можливість максимально оптимізувати процес збору даних для дорожньо-транспортної сфери та, завдяки зручному інтерфейсу, значно полегшити аналіз отриманої інформації.

# 1. ЗАДАЧА ЗБОРУ ДАНИХ ТРАНСПОРТНОЇ ТЕЛЕМАТИКИ

Основною метою дипломної роботи є створення системи зчитування даних з сенсорів смартфону та аналізу інформації для подальшого її використання у сфері транспортної телематики.

Опис моделі роботи майбутньої системи:

Програмне забезпечення має виконувати такі задачі, як збір інформації за допомогою сенсорів мобільного телефону та для подальшого використовування цієї інформації, а саме: аналізу стану доріг, аналізу трафіку на дорогах, побудови оптимальних маршрутів.

Програмне забезпечення призване надати можливість адміністраторам системи використовувати отриману інформацію для аналізу. Користувачі, за допомогою яких буде відбувається збір інформації, мають отримувати результати аналізу вищезгаданих даних.

Клієнт системи (а саме водій) буде використовувати мобільний додаток як дорожній навігатор, який буде надавати корисну інформацію для нього, у нашому випадку стан доріг, а додаток має паралельно зчитувати інформацію з усіх можливих сенсорів мобільного телефону та відправляти дані на сервер. В свою чергу, адміністратору системи потрібно надати змогу зайти веб-додаток та отримати доступ до зібраної користувачами інформації та провести її аналіз використовуючи зручний веб-інтерфейс. Також адміністратору системи потрібно надати доступ до розширеної версії мобільного додатку у якому можна переглядати, у режимі реального часу, значення усіх можливих показників, що будуть зчитані з сенсорів телефону. Адміністратору потрібно надати більше можливостей під часу збору даних.

Таким чином, можна виділити наступні задачі:

1. Створити додаток користувача (базова версія).
2. Створити додаток адміністратора (розширена версія).

3. Реалізувати зв'язок інформаційних потоків між сервером бази даних та клієнтськими додатками через API сервер.

4. Розробити модель бази даних, яка б містила інформацію про показники сенсорів телефону, отриману від мобільних додатків, дані про стан дорожнього покриття, а також інформацію про користувачів системи.

Функції підсистем:

Мобільний додаток клієнта

Має надати можливість клієнту:

- переглянути карту
- переглянути своє місцеположення
- переглянути швидкість пересування
- переглянути інформацію стану доріг, відносно свого положення (дорогу поруч з користувачем)
- прокласти маршрут до відповідного місця, та інформацію про цей маршрут
- переглянути завантаженість доріг

Також додаток має надати змогу зчитувати та фільтрувати інформацію з сенсорів телефону під час руху та відправляти її на сервер.

Мобільний додаток адміністратора

Має надати можливість адміністратору:

- переглянути карту
- переглянути своє місцеположення
- переглянути інформацію стану доріг відносно свого положення (дорогу поруч з користувачем)
- переглянути показники сенсорів у режимі реального часу
- побачити графічну інтерпретацію показників сенсорів
- розширений функціонал для запису показників сенсорів

Також додаток має надати змогу зчитувати та фільтрувати інформацію, отриману від сенсорів телефону під час руху та відправляти її на сервер.

Веб-додаток адміністратора

Має надати можливість адміністратору:

- переглянути стан системи
- переглянути список запитів показників сенсорів
- переглянути деталі запису показників сенсорів з відповідними інструментами для цього
- ручне завантаження записів значень сенсорів

Також веб-додаток повинен бути захищений системою авторизації

API сервер

Має надати можливість:

- керувати авторизацією додатків системи
- отримувати дані з мобільного додатку та зберігати у бд
- робити пре-обробку отриманих даних: фільтрацію шумів та реорієнтацію значень
- робити пост-обробку – аналізувати отримані дані з метою отримання стану дороги на учаснику на якому були зібрані дані
- відправляти дані БД в усі додатки

Вхідна інформація: показники сенсорів.

Вихідна інформація: схеми, діаграми, інформація про стан доріг, інформація про завантаженість дороги, маршрут до місця призначення з інформацією про цей маршрут.

Користувачі системи: Водії автомобілів та будь-які люди, діяльність яких пов’язана з дорожньою інфраструктурою.

# 2. МЕТОДИ ТА ЗАСОБИ ТРАНСПОРТНОЇ ТЕЛЕМАТИКИ

## 2.1. Транспортна телематика

З другої половини XX століття в США та Європі почали впроваджувати у транспортні засоби спеціальні системи, з метою:

- збільшення ефективності транспортних процесів
- збільшення безпеки транспортних процесів
- надання інформації учасникам дорожнього руху та центрам управління про ситуацію на дорозі

Дані системи отримали назву – системи транспортної телематики.

Термін «Телематика» – це похідна від слів «телекомунікація» та «інформатика». Відповідно, поняття «транспортна телематика» охоплює область використання можливостей телекомунікаційних технологій та інформатики для вирішення проблем транспорту.

«Телематичні системи» – це комплекс взаємопов'язаних автоматизованих систем, що вирішують завдання управління дорожнім рухом, моніторингу та управління роботою всіх видів транспорту, інформування громадян і підприємств про організацію транспортного обслуговування на території регіону[1].

На сьогоднішній день транспортна телематика працює над досягненням наступних цілей:

- збільшення безпеки дорожнього руху
- збільшення пропускної спроможності і оптимізації вулично-дорожньої мережі
- зниження ризиків виникнення надзвичайних ситуацій та їх наслідків
- збільшення ефективності транспортної системи
- збільшення інформованості учасників дорожнього руху
- оптимізації роботи дорожніх служб

В даний час впровадження комплексних ІТС (Інтелектуальні транспортні системи) об'єднує телекомунікаційні та інформаційні технології з організацією руху транспортних потоків так, щоб підвищити пропускну здатність існуючої транспортної інфраструктури, а також підвищити безпеку та поліпшити екологію транспортних систем. Транспортна телематика при цьому є елементом технічного забезпечення основних функціональних і системних компонентів ІТС[1].

## 2.2. Смартфон як частина інтелектуальних транспортних систем

На сьогоднішній день мобільний телефон став невід'ємним атрибутом життя сучасної людини. Ця платформа вже поглинула та продовжує поглинати велику кількість інших технологій з різноманітніших сфер діяльності, за рахунок постійного розвитку даної платформи.

Смартфони використовують програми для об'єднання можливостей комп'ютера з мобільністю смартфону. Смартфон став визначати соціальне життя в нинішньому часі. Вже 2016 році було продано більш ніж півтора мільярда одиниць. Постійно зростаючий інтерес до смартфона як універсального вимірювального пристрою має декілька пояснень.  Одне з них – це наявність великої кількості вбудованих датчиків, можливість бездротової передачі даних та наявність засобів для соціальної взаємодії.

Використання смартфонів для збору даних підпадає під загальне поле телематики. Посилаючись на послуги, в яких використовуються телекомунікації для передачі інформації, яка надається сенсорами. Як приклад можна привести телематику транспортних засобів або розумних будівель. Завдяки постійно зростаючому світовому впровадженню смартфонів, автомобільна та навігаційна індустрія отримала нові способи збору даних, що, в свою чергу, принесло користь водіям, власникам транспортних засобів та суспільству в цілому. Величезна кількість здійснених проектів, як в наукових колах, так і в різних галузях

виробництва, ілюструє значний потенціал телематики на основі смартфонів і заклала стійку основу для майбутніх впроваджень на масовому ринку.

Є кілька причин віддавати перевагу телематиці на основі смартфона перед реалізаціями, які використовують виключно сенсори, що фіксуються автомобілем. Завдяки зростанню права власності на смартфони, рішення на основі смартфонів, як правило, масштабовані, модернізовані та дешеві. Крім того, смартфони є основними платформами для надання миттєвої зворотної інформації з водіями за допомогою аудіовізуальних засобів, які дозволяють плавно інтегрувати телематичні послуги з існуючими соціальними мережами. Більше того, цикли оновлення та розвитку смартфонів значно коротші, ніж у транспортних засобів, тому смартфони можуть часто пропонувати та закликати до використання нових технологій.

У той же час, поріг входження для кінцевого споживача досить низький. Як правило, обмежуються встановленням мобільного додатку з відповідного агрегатору додатків в залежності від операційної системи.

Однак використання смартфонів у телематиці транспортних засобів також викликає кілька проблем. Вбудовані сенсори смартфонів, як правило, низької якості, бо їх не вибирали та не розробляли з урахуванням телематичних додатків автомобілів. Як результат, кінцеві алгоритми повинні обов'язково покладатися на статистичні моделі шуму, враховуючи неточність сенсорів смартфона. Крім того, в залежності від користувача, смартфон фіксується у різних положеннях щодо транспортного засобу, що ускладнює інтерпретацію даних, залежних від орієнтації датчиків, таких як акселерометри, гіроскопи та магнітометри. Ще одне питання - це заряд акумулятора. Додаток, який постійно збирає, зберігає, обробляє та відправляє дані, споживає досить багато енергії, тому виникає потреба у рішеннях, які би обмежили та збалансували продуктивність та енергоефективність. І нарешті, потрібно зазначити, що смартфон, створено для взаємодії з людиною, а не за автомобілем. За рахунок цього смартфон може задовольняти більшість потреб водія, не втрачаючи функціональних можливостей телематичних служб на основі смартфону. Однак це також може бути недоліком, наприклад, поліси

автомобільного страхування, які збирають дані для страхових цілей, слідують безпосередньо за транспортним засобом, де водій не зможе фальсифікувати ці дані.

## 2.3. Сенсори смартфона та вимірювання їх сигналів

Вимірювання сигналів від сенсорів смартфонів є складним завданням через різні властивості сенсорів у різних моделях смартфонів, відмінності у їх розмірах, вазі, довжині та системах підвіски автомобіля. Також існує різниця у русі транспортного засобу, викликаному аномаліями дорожнього покриття, а саме: довжина, глибина та форма вибоїн на дорожньому покритті та вплив кривизни дороги на швидкість руху різних транспортних засобів[1].

Різні транспортні засоби, що проїжджають через конкретну вибоїну, не створюють ідентичну схему сигналу. Крім того, швидкість руху транспортного засобу впливає на швидкість зміни сенсорів руху, що призводить до різних моделей реакції вібрації на будь-яку аномалію дорожнього покриття.

Сучасні смартфони мають велику кількість типів вбудованих сенсорів. Одні з них є апаратними (фізичними), а інші – програмними (віртуальними)[2].

Апаратні сенсори – це фізичні вбудовані сенсори, такі як акселерометр, гіроскоп, магнітометр, датчик світла, температура, тощо. А програмні сенсори використовують дані від одного або декількох апаратних сенсорів і обчислюють значення в режимі реального часу, наприклад: лінійне прискорення, обертання, гравітація та інші.

Також сенсори смартфонів можна класифікувати на три різні типи: сенсори руху, сенсори положення, та екологічні сенсори.

Сенсори руху підходять для контролю руху та вібрації пристрою, нахилу, тремтіння, обертання або гойдання. Рухи можуть безпосередньо відображати взаємодію користувачів, з програмою, наприклад коли користувач керує автомобілем або керує об’єктом у мобільній грі.

Сенсори положення підходять для визначення фізичного положення пристрою в системі координат місцевого рівня. Наприклад, сенсор геомагнітного поля в поєднанні з сенсорами акселерометра може визначити положення пристрою відносно системи координат місцевого рівня.

Екологічні сенсори вимірюють екологічні властивості навколишньої території, такі як температура, вологість, атмосферний тиск та освітленість[2].

Для кожної програми на основі смартфона різні комбінації сенсорів можуть використовуватися залежно від бажаних критеріїв застосування. За допомогою цих сенсорів можна розробити додаток для аналізу дорожнього покриття, виявлення аномалій.

За показниками сенсорів руху можна відслідковувати  похитування або нахил в рухомому транспортному засобі, викликаного аномаліями дорожнього покриття.

Для визначення поточного місця розташування смартфона (широту, довготу), орієнтацію напрямку переміщення та швидкість руху можна використовувати GPS (Global Positioning System) та Wi-Fi.

У таблиці 2.1 наведено перелік усіх сенсорів, які зазвичай зустрічаються у смартфонах та які можна використовувати для аналізу дорожнього покриття.

*Таблиця 2.1*

| Назва | Тип | Одиниця вимірювання | Опис |
|---|---|---|---|
| Акселерометр | Апаратний | $м/с^2$ | Вимірює силу прискорення |
| Гіроскоп | Апаратний | $рад/с^2$ | Вимірює швидкість обертання пристрою |
| Магнітометр | Апаратний | μT | Вимірює навколишнє магнітне поле |
| Гравітація | Програмний | $м/с^2$ | Вимірює силу тяжіння |
| Обертання | Програмний | радіани | Вимірює орієнтацію |

| | | | |
|---|---|---|---|
| | | | пристрою |
| GPS | Апаратний | градуси | Отримує інформацію про місцезнаходження |

## 2.4. Можливі підходи для виявлення дорожніх аномалій

Виділяють три основні підходи для автоматизації моніторингу дорожнього покриття:

- базований на 3D-реконструкції
- базований на відео спостереженні
- базований на вібраціях автомобіля

3D-підхід покладається на лазерне 3D-сканування для створення точних цифрових моделей поверхні. Тобто, 3D-сканування – це процес аналізу об'єкта чи середовища в реальному світі для збору даних про його форму та зовнішній вигляд. Потім зібрані дані можуть бути використані для побудови цифрових 3D-моделей.

3D-сканер може працювати на багатьох різних технологіях, кожна з яких має свої обмеження, перевага та витрати. Із значних недоліків оптична технологія може зіткнутися з багатьма труднощами, наприклад, з блискучими або прозорими предметами.

У такому підході лазерний 3D-сканер використовує відбиті лазерні імпульси, які створюють точні 3D-цифрові моделі існуючих об'єктів, наприклад, аномалії дорожнього покриття. За рахунок точної моделі та візуалізації місцевості, відкривається багато можливість для найкращого аналізу стану дороги. Однак вищезазначений підхід вимагає багато лазерних сканерів і вимагає значних затрат при моніторингу масштабних дорожніх мереж[3].

Підхід на основі відео спостереження покладається на аналіз обробки зображень відеокамер, на основі яких відбувається знаходження текстури дороги та порівняння з вже знятими фотографіями певних дорожніх аномалій. Така система використовує зображення з прив'язкою до місцевості, зняті камерою або

відеосистемою, яка встановлена на рухомому транспортному засобі. Будь-які підозрілі об'єкти на дорожньому покритті, включаючи вибоїни та тріщини, можуть бути автоматично виявлені із зібраних відеозображень. Незважаючи на те, що цей підхід є більш економічний в порівнянні з першим, який базується на 3D-реконструкції, він залежить від певних умов навколишнього середовища, таких як освітлення, тіньовий вплив, тощо[4].

З використанням підходу, що базується на вібрації, дорожні події виявляються за швидкістю вібрацій рухомих транспортних засобів, захоплених датчиками руху (наприклад, акселерометрами або гіроскопами). Теоретично транспортний засіб при переході через будь-яку аномалію дорожнього покриття, наприклад, вибоїну, тріщину, люк, буде вібрувати більше, ніж при переході по гладких дорожніх покриттях. За останні кілька років збір даних на основі смартфонів став важливою доповненою методикою виявлення аномалій дорожнього покриття з метою спостереження за поверхнею ділянок дороги або поверхнею велосипедних та пішохідних смуг. Очікується, що датчики, є новою областю, в якій вимірювання на основі смартфона здаються особливо привабливими, оскільки вони широко використовуються[5].

Вимірювання та аналіз сигнальних датчиків руху від різних типів мобільних пристроїв можуть розходитися залежно від багатьох факторів, включаючи характеристики та якість сенсорів, положення пристрою смартфона, систему підвіски автомобіля та швидкість.

Дипломна робота пропонує комплексний огляд існуючих досліджень, у сфері виявлення дорожніх подій за допомогою сенсорів смартфону, та аналізу дорожнього покриття.

Було прийнято рішення взяти за основу підхід, що базується на вібраціях автомобіля, на базі сенсорів смартфона. Підставою для цього стало значне поширення мобільних пристроїв у суспільстві, та наявність потрібних сенсорів для подальшого аналізу та досліджень.

## 2.5. Виявлення аномалій

На етапі обробки використовуються попередньо зібранні, проаналізовані та приведені до зручного для роботи вигляду значення сенсорів, для виявлення аномалій дорожнього покриття. Значення акселерометра найбільш сприятливі до аналізу дорожніх аномалій серед інших сенсорів, за рахунок відстеження прискорення об'єкта. При пересуванні по нерівній дорозі автомобіль створює незначні коливання у просторі, завдяки чому є можливість виявити аномальні зміни сенсора. Для проведення такого аналізу існують три базові підходи:

Перший підхід засновано на порівнянні заздалегідь експериментально визначених та проаналізованих порогових значеннях показників сенсорів, з новими поточними значеннями.

Другий підхід базується на машинному навчанні, він використовує вже більш вдосконалені методики та алгоритми для виявлення дорожнього покриття. Наприклад, можна використовувати метод опорних векторів, який є методом аналізу даних для класифікації за допомогою моделей з керованим навчанням. Для заданого набору тренувальних зразків, які було віднесено до одної або іншої категорії (добре або погане дорожнє покриття), алгоритм будує модель, яка відносить нові зразки до однієї чи іншої категорії. Модель даного методу є представленням показників як точок у просторі відображених таким чином, що їх з окремих категорії розділяє гіперплощина з максимальним проміжком у цьому просторі[6].

Третій підхід основано на алогоритмі динамічної трансформації тимчасової шкали (DTW-алгоритм). Цей алгоритм дозволяє знайти відповідність між двома часовими рядами і переважно застосовується в дослідженнях розпізнавання мовлення. Алгоритм порівнює вхідні дані сигналів із заздалегідь заданими шаблонами та вимірює схожість між наборами даних.

## 2.6. Опис аналогічних систем у сфері транспортної телематики

### Телематичний комплекс StarLine

StarLine компанія, яка розробляє телематичний комплекс для застосування на автомобільному транспорті з системами авторизації власника. У його комплект включена уся необхідна електроніка для створення надійного охоронного комплексу та аналізу вузлів і агрегатів транспортного засобу.

Також є мобільний додаток для зручного керування автомобілем. Який допомагає виявити помилки робити бортового обладнання та передати цю інформацію користувачу, моніторинг стану автомобіля, моніторинг та контроль за пересуванням, а також місцем знаходження автомобіля, авторизація володаря автомобіля, що сприяє захисту від його викрадення[7].

Переваги:

- захист від викрадення автомобіля
- моніторинг стану автомобіля
- моніторинг та контроль пересування та знаходження автомобіля
- дистанційне керування автомобілем
- аналітика поведінки водія на дорозі

Недоліки:

- телематичний комплекс достатньо дорого коштує
- не кожен автомобіль підтримує телематичний комплекс
- більша частина інформації у приватному доступі

### GoogleMaps

Набір додатків побудованих на картографічному сервісі та технологіях наданих компанією Google. Має повний доступ до карт, за рахунок чого має можливість інтегрувати сторонні сервіси, прокладати маршрути, та має доступ до локації користувача завдяки чому може робити аналізи стосовно цього, наприклад, аналіз трафіку на дорозі[8]. Знаходиться у відкритому доступі та вважається одним з найголовніших елементів при побудові додатків для навігації.

Переваги:

- одна з найбільш деталізованих карт
- містить багато сторонніх інтеграцій (розклад маршрутів, опис закладів)
- може прокладати маршрути
- трафік на дорозі (затори і т.п.)
- вільний у користуванні (безкоштовний)

Недоліки:

- додаток не орієнтовано на конкретну групу споживачів (водіїв), тому він не може задовільнити усі потреби водія.
- більша частина інформації у приватному доступі

**Waze**

Безкоштовний мобільний додаток навігації орієнтований на водіїв, що дозволяє відслідковувати стан на дорогах у режимі реального часу, прокладати оптимальні маршрути, інформувати інших користувачів про місцезнаходження радарів швидкості, перешкод та поліції, попереджувати про зміну дорожніх обставин, спілкуватись з іншими користувачами на карті. Карти у Waze заповнюються і створюються за допомогою користувачів[9].

Переваги:

- карта деталізується за допомогою користувачів
- може прокладати оптимальні маршрути
- містить інформацію про стан дороги

має спідометр та зручний інтерфейс для водія (наприклад, підтримує голосові команди)

- має можливість інформування оточуючих про стан дороги

Недоліки:

- процес створення та фіксування інформації про стан роботи не автоматизовано, уся інформація надається напряму користувачами
- більша частина інформації у приватному доступі

# 3. ЗАСОБИ РОЗРОБКИ

## 3.1. Вибір технологій та їх обґрунтування

Систему було реалізовано за допомогою мов програмування Swift та JavaScript. Серверна частина веб-додатку(API) було реалізовано за допомогою мови програмування Swift та фреймворка Vapor. Графічна частина веб-додатку була реалізована за допомогою мови програмування JavaScript, на сьогодні це найбільш оптимізована та найпопулярніша мова для розробки графічної частини веб-додатків, яка є монополістом у даній галузі. Мобільний додаток було реалізовано за допомогою мови програмування Swift на платформі iOS, яка також є основною для даної галузі, виключення Objective-C, але Swift було впроваджено, як сучасну, оптимізовану заміну Objective-C. Додаток створено на основі фреймворку UIKit. Заміною UIKit, міг би стати фреймворк SwiftUI, але після дослідження було виявлено, що він ще достатньо молодий та не має більшості необхідних інструментів.

Для взаємодії з сенсорами телефону було використано наступні бібліотеки:

- CoreLocation – бібліотека для роботи з GPS
- CoreMotion – бібліотека для роботи з сенсорами телефона (акселерометр, гіроскоп, магнітометр)

## 3.2. Мова програмування Swift

Багатопарадигмова компільована статична мова програмування з відкритим кодом, що розробляється компанією Apple. Вона побудована на базі LLVM компілятора, та компілюється у машинний код.

Swift з'явився у 2010 році, як заміна Objective-C, як більш сучасна та стійкіша мова до помилкового коду мова програмування, для розробки iOS, macOS, watchOS,

tvOS додатків. Тобто для написання додатків на операційних системах розроблених компанією Apple[10].

Swift уособлює кращі переваги таких мов як Objective-C, C, Rust, Haskell, Ruby, Python, C#. Але відрізняється підходом до контролю пам'яті, використовує засоби підрахунку посилань на об'єкти, а також LLVM автовекторизацію.

## 3.3. Мова програмування JavaScript

JavaScript це динамічна багатопарадигмова прототипна мова програмування. Підтримує багато стилів програмування, а саме: об'єктно-орієнтований, функціональний, імперативний. Та вважається реалізацією стандарту ECMA Script. Найбільш широке поширення має в браузерах для надання інтерактивності веб-додаткам. JavaScript позиціонується як прототипна, скриптова, асинхронна мова програмування[11].

## 3.4. Середа розробки xCode

xCode – це інтегрована середа розробки(IDE) від компанії Apple. Яка дозволяє створювати програмне забезпечення, в першу чергу, для таких мов як Swift, Objective-C, C, а також розуміє і надає візуальну допомогу при написанні коду таких мов як JavaScript, SQL, Ruby та інших мовах[12].

xCode має безліч корисних інструментів для аналізу роботи та оптимізації iOS додатків, наприклад, інструменти для аналізу використання оперативної пам'яті та пошуку витіку пам’яті (memory leak), інструменти для аналізу енерговитрат пристрою під час використання вашого додатку, інструменти для аналізу відображення графічного інтерфейсу та безліч інших. Завдяки всім цім інструментам розробник має можливість знайти проблемні ділянки додатку та оптимізувати їх.

## 3.5. Фреймворк UIKit

UIKit це iOS фреймворк який забезпечує необхідні інфраструктуру для iOS додатків, надає архітектуру для реалізації графічного інтерфейсу, а також основний цикл запуску, необхідний для взаємодії між користувачем, операційною системою та додатком.

UIKit для відображення графічних елементів використовує Auto-Layout. Auto-Layout динамічно обчислює розмір і положення всіх графічних елементів у вашій ієрархії представлень, покладаючись на обмеження, розміщених на цих представленнях. Наприклад, ви можете обмежити кнопку так, щоб вона була горизонтально розташована по центру зображення, і щоб верхній край кнопки завжди залишався на 8 пунктів нижче нижньої частини зображення. Якщо розмір або положення перегляду зображення змінюються, положення кнопки автоматично перераховується[13].

Такий підхід до проектування на основі обмежень дозволяє створювати інтерфейси користувача, які динамічно реагують як на внутрішні, так і на зовнішні зміни.

## 3.6. Мова програмування SQL

В перекладі з англійської мови Structured Query Language – мова структурованих запитів. SQL являє собою декларативну мову програмування, що використовується для формування запитів до реляційних баз даних, а також для оновлення і керування ними.

Мова структурних запитів використовується розробниками для створення схеми бази даних, а також її модифікації або видалення. SQL – це лише мова програмування, вона не є а ні системою керування базами даних, а ні окремим програмним продуктом[14].

## 3.7. СУБД MySQL

MySQL, на відміну від SQL, це не мова, а вільна система керування реляційними базами даних. Використовується, як інструмент для створення динамічних веб-сторінок [14]. MySQL гарно підтримується різноманітними мовами програмування.

## 3.8. Веб-фреймворк Vapor

Vapor - це веб-фреймворк з відкритим кодом, який на написано на базі мови програмування Swift. Vapor використовує для роботи з інтернет мережею офіційний фреймворк Swift NIO, також створений компанією Apple. Він може бути використаний для створення API RESTful, веб-додатків та додатків у режимі реального часу за допомогою WebSockets. Також Vapor має ORM, мову шаблонів Leaf та бібліотеки полегшення автентифікації та авторизації користувачів[15].

Фреймворк Leaf - шаблонна мова для створення динамічних HTML-сторінок для веб-сайту або для створення електронних листів, що надсилаються з API.

Фреймворк Fluent - швидкий та безпечний у використанні фреймворк ORM, створений для Swift, який забезпечує основу для побудови інтеграції баз даних.

# 4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 4.1. Структура програмного забезпечення.

Програмний продукт складається з двох мобільних додатків, API серверу, веб-додатку, бази даних та був побудований на клієнт серверній архітектурі з різними рівнями доступу. В додатку є два типи користувачів:

- Адміністратор – має доступ до веб-додатку, розширеної версії мобільного додатку та до всіх даних в БД.
- Користувач – має доступ лише до базової версії мобільного додатку

На рисунку 4.1 подано діаграму розгортання розробленої системи.

Діаграма розгортання системи показує обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Діаграма розгортання в UML нотації моделює фізичне розгортання артефактів на вузлах.

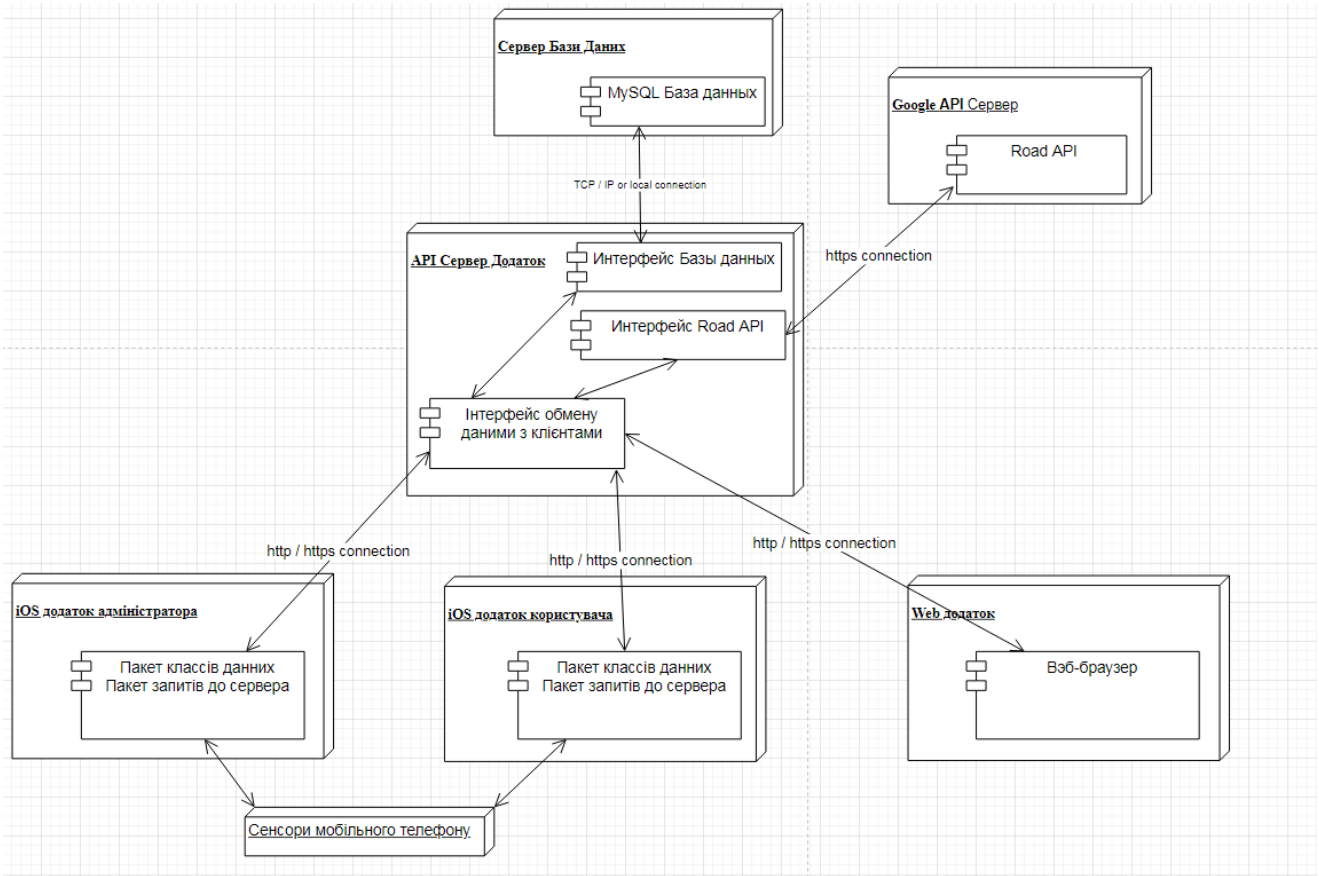


Рисунок 4.1 – Діаграма розгортання системи.

## 4.2. Опис бази даних

Для реалізації поставленої задачі було розроблено базу даних засобами мови програмування MySQL, яка складається з 12 таблиць.

Таблиця “Прискорення” містить інформацію про зчитану інформацію з акслерометра, що включає в себе індекс, та прискорення по трьом осям (таблиця 4.1.).

Таблиця 4.1. Структура таблиці “Прискорення”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремет, унікальне поле в таблиці |
| Значення осі X | double | Прискорення по осі X |
| Значення осі Y | double | Прискорення по осі Y |
| Значення осі Z | double | Прискорення по осі Z |

Таблиця “Розташування у просторі” (таблиця 4.2.) містить зчитану інформацію отриману через акселерометр, гіроскоп та магнітометр, орієнтація аерокосмічного апарата щодо інерціальної системи відліку.

Таблиця 4.2. Структура таблиці “Розташування у просторі”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Значення pitch | double | Поперечна вісь |
| Значення yaw | double | Вертикальна вісь |

| | | |
|---|---|---|
| Значення roll | double | Поздовжня вісь |

Таблиця “Вектор Гравітації” (таблиця 4.3.) містить зчитану інформацію з акслерометра та магнітометра, що являє собою нормований вектор сили тяжіння.

Таблиця 4.3. Структура таблиці “Вектор Гравітації”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Значення осі X | double | Сила тяжіння відносно осі X |
| Значення осі Y | double | Сила тяжіння відносно осі Y |
| Значення осі Z | double | Сила тяжіння відносно осі Z |

Таблиця “Швидкість обертання” (таблиця 4.4.) містить зчитану інформацію з акслерометра та магнітометра, що являє собою нормований вектор сили тяжіння.

Таблиця 4.4. Структура таблиці “Швидкість обертання”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Значення осі X | double | Швидкість обертання по осі X |
| Значення осі Y | double | Швидкість обертання по осі Y |
| Значення осі Z | double | Швидкість обертання по осі Z |

Таблиця “Локація” (таблиця 4.5.) мастить зчитану інформацію з GPS.

Таблиця 4.5. Структура таблиці “Локація”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Довгота | double | Довгота |
| Широта | double | Широта |
| Похибка місця знаходження | double | Швидкість обертання по осі Z |
| Курс пересування | double | Напрямок пересування у радіанах |
| Швидкість | double | Швидкість пересування (км\год) |
| Дата коли дані були отримані | datetime | Дата коли дані були отримані |

Таблиця “Елемент запису сесії” (таблиця 4.6.) містить зчитану з усіх сенсорів телефону.

Таблиця 4.6. Структура таблиці “Елемент запису сесії”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Ідентифікатор сесії | int | Ідентифікатор для зв'язку з таблицею “Сесія” |
| Ідентифікатор прискорення | int | Ідентифікатор для зв'язку з таблицею “Прискорення” |
| Ідентифікатор розташування у просторі | int | Ідентифікатор для зв'язку з таблицею “Розташування у просторі” |
| Ідентифікатор вектору гравітації | int | Ідентифікатор для зв'язку з таблицею “Вектор гравітації” |
| Ідентифікатор | int | Ідентифікатор для зв'язку з таблицею |

| | | |
|---|---|---|
| швидкості обертання | | “Швидкість обертання” |
| Ідентифікатор локації | int | Ідентифікатор для зв'язку з таблицею “Локація” |
| Час коли дані були отримані | datetime | Час коли дані були отримані |
| Часова мітка відносно сесії запису | double | Час пройдений відносно початку запису |

Таблиця “Сесія” (таблиця 4.7.) містить зчитану з усіх сенсорів телефону.

Таблиця 4.7. Структура таблиці “Сесія”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Назва | string | Назва для класифікації подій на даній ділянці |
| Частота оновлення даних сенсорів | double | Частота з якою дані сенсорів оновлюються |
| Час коли почалась сесія | datetime | Час коли почалась сесія |
| Ідентифікатор користувача від кого отримано запис | int | Ідентифікатор для зв'язку з таблицею “Користувач” |
| Ідентифікатор чи проіндексовані дані | tinyint | Флаг для перевірки чи проаналізовані дані |

Таблиця “Користувач” (таблиця 4.8.) містить дані про користувача.

Таблиця 4.8. Структура таблиці “Користувач”

| Назва поля | Тип поля | Призначення |
|---|---|---|

| | | |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Електрона пошта | double | Електрона пошта, використовується для авторизації користувача |
| Зашифрований пароль | double | Зашифрований пароль, використовується для авторизації користувача |
| Роль | dobule | Роль користувача у системі |

Таблиця “Сесія користувача” (таблиця 4.9.) містить дані сесії авторизації.

Таблиця 4.9. Структура таблиці “Сессія користувача”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Значення ідентифікатору | string | Значення ідентифікатору для підтвердження сессії |
| Ідентифікатор користувача | int | Ідентифікатор для зв'язку з таблицею “Користувач” |
| Дата до якої дійсна сесія | datetime | Дата після якою сессія вважається недійсною |

Таблиця “Вузол графа дороги” (таблиця 4.10.) містить вузол графа для побудови графа доріг.

Таблиця 4.10. Структура таблиці “Вузол графа дороги”

| Назва поля | Тип поля | Призначення |
|---|---|---|

| | | |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Широта | double | Широта |
| Довгота | double | Довгота |

Таблиця “Ребро графа стану доріг” (таблиця 4.11.) містить ребро графа для побудови графа стану доріг.

Таблиця 4.11. Структура таблиці “Ребро”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Ідентифікатор першого вузла | double | Ідентифікатор для зв'язку з таблицею “Вузол графа дороги” |
| Ідентифікатор другого вузла | double | Ідентифікатор для зв'язку з таблицею “Вузол графа дороги” |
| Оцінка стану дороги | double | Оцінка стану дороги |
| Дата попереднього оновлення | datetime | Дата попереднього оновлення, потрібно для перерахунку оцінки при наступній обробці |
| Кількість обробок | int | Кількість обробок, потрібно для перерахунку оцінки при наступній обробці |
| Вага обробок | double | Вага обробок, потрібно для перерахунку оцінки при наступній обробці |

Таблиця “Історія дій в системи” (таблиця 4.12.) містить історію дій в системі.

Таблиця 4.12. Структура таблиці “Історія дій в системи”

| Назва поля | Тип поля | Призначення |
|---|---|---|
| Ідентифікатор | int | Автоінкремент, унікальне поле в таблиці |
| Ідентифікатор виконавця | double | Ідентифікатор для зв'язку з таблицею “Користувач” |
| Опис | double | Опис події в системі |

На рисунку 4.2 подано схему бази даних зі зв’язками.

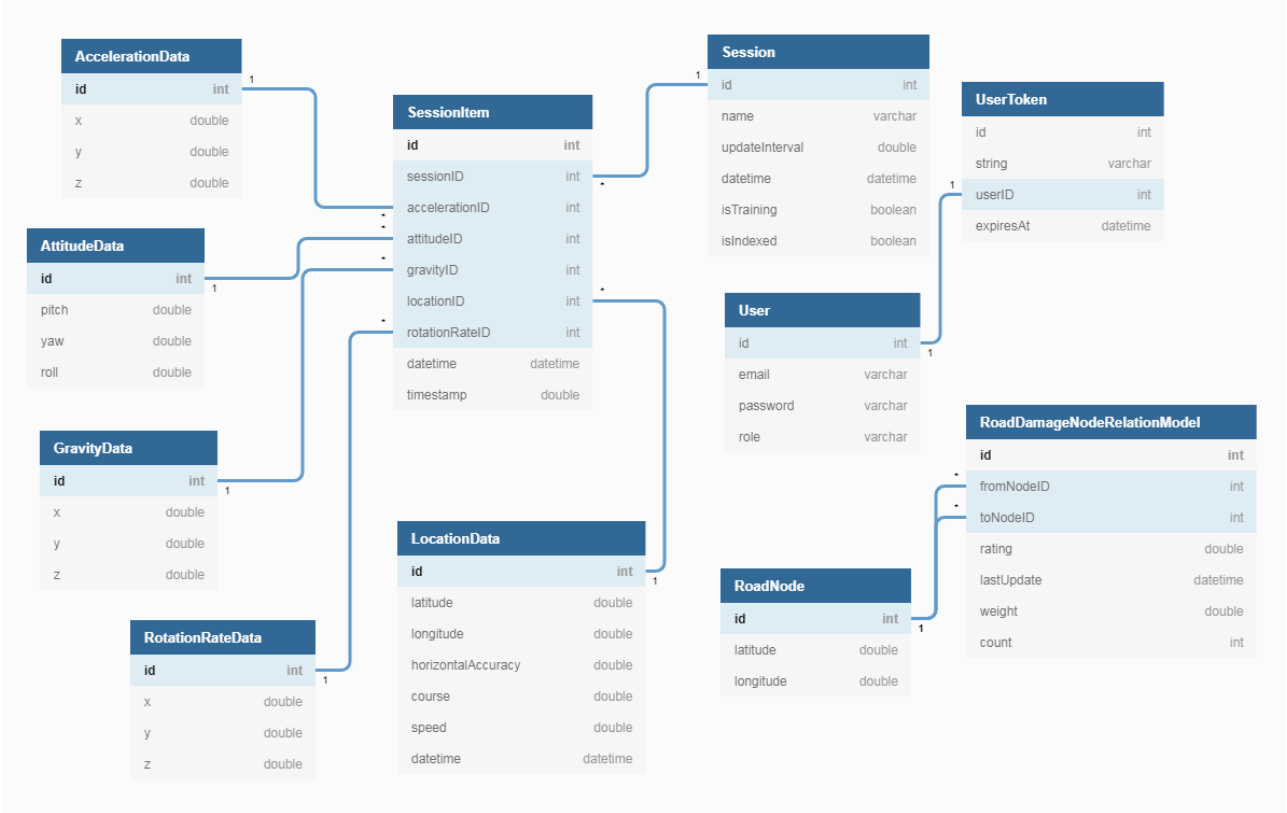


Рисунок 4.2 – Схема бази даних.

Схема бази даних представляє собою структура системи, що розроблюється. База даних описана формальною мовою, яка підтримується системою керування баз даних (СКБД).

## 4.3. Діаграма прецедентів системи

На рисунку 4.3 подано діаграму прецедентів системи.


Рисунок 4.3 – Діаграма прецедентів системи

## 4.4. Розміщення пристрою

Перша й одна з ключових проблем полягала в тому, що розміщення пристрою зчитування всередині транспортного засобу може вплинути на якість збору даних. Щоб дослідити цю проблему, ми розмістили пристрої в трьох кріпильних місцях автомобіля, а саме:

- на лобовому склі
- на передній панелі автомобіля
- біля бортового комп'ютеру

На рисунках 4.4 – 4.6 показано побачити дані акселерометру та гіроскопу з різних кріпильних положень. Остання позиція була перевірена для визначення, наскільки важливо щоб пристрій був міцно і нерухомо для коректного збору інформації.

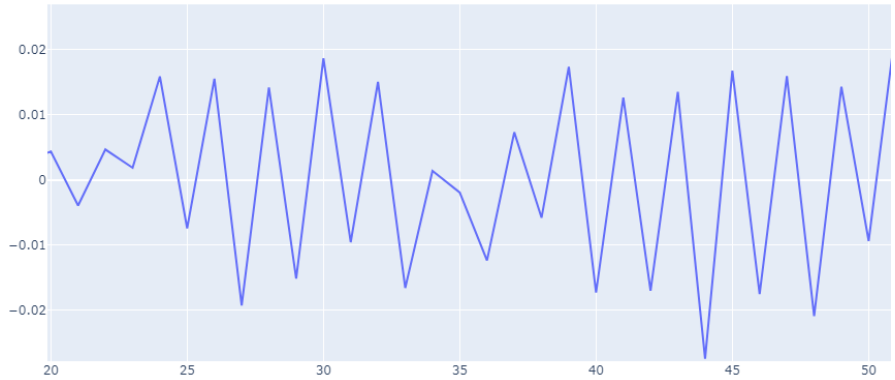


Рисунок 4.4 – Прискорення по осі Y, коли пристрій прикріплено щільно на лобовому склі

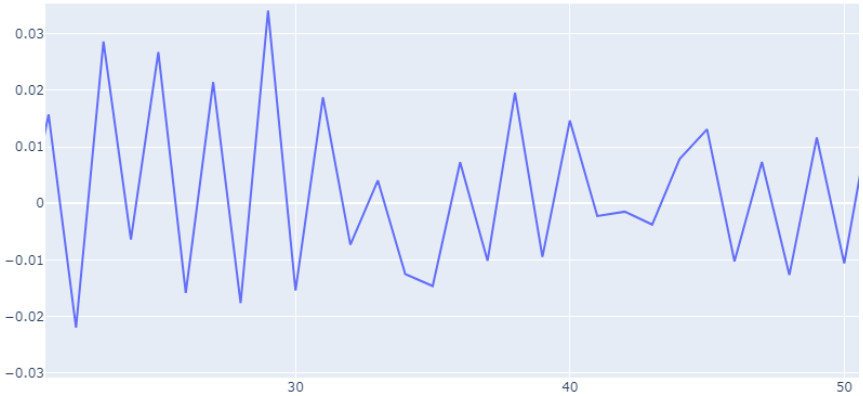


Рисунок 4.5 – Прискорення по осі Y, коли пристрій прикріплено щільно на передній панелі автомобіля

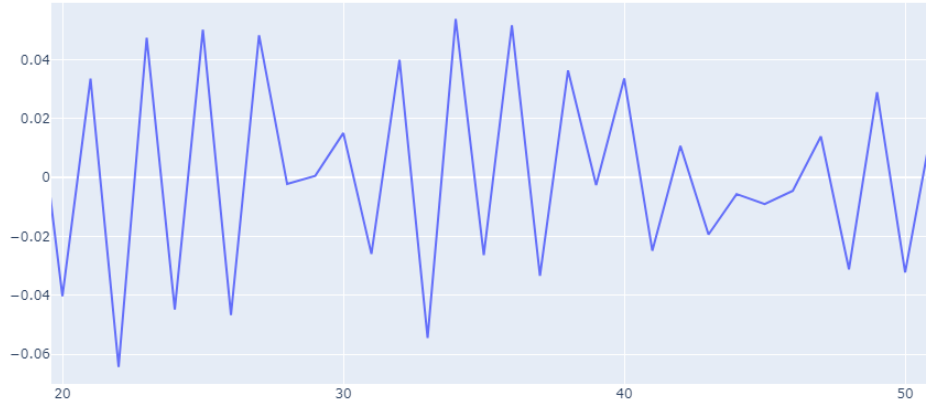


Рисунок 4.6 – Прискорення по осі Y, коли пристрій прикріплено не щільно біля бортового комп'ютеру

Сигнали з панелі приладів та лобового скла, досить схожі, тоді як акселерометр, прикріплений до комп'ютера, дав непередбачувані великі значенні. Як результат, було вирішено прикріпити зчитувальний пристрій до передньої панелі автомобіля, тому що таке місце положення не заважає водію, на відміну від лобового скла.

## 4.5. Геолокація користувача (GPS)

Один з найважливіших показників для подальшого використання здобутої інформації це прив'язка до місцевості. Після досліджень було встановлено що допустиме відхилення для подальшого аналізу становить 3,5 – 6 метрів. При більшому відхиленні буде важко працювати з інформацією закріпленою до цього місцеположення. При звичайних умовах відхилення становить в середньому 4 метра, що є достатньою мірою для подальшого аналізу та відповідає типовим помилкам вимірювань від сучасних GPS-приймачів на вулиці. Через роботу GPS з похибкою, при зчитуванні даних сенсорів у автомобілі, який рухався по трасі H15 локація буде виглядати, як показано на рисунках 4.7 – 4.8.

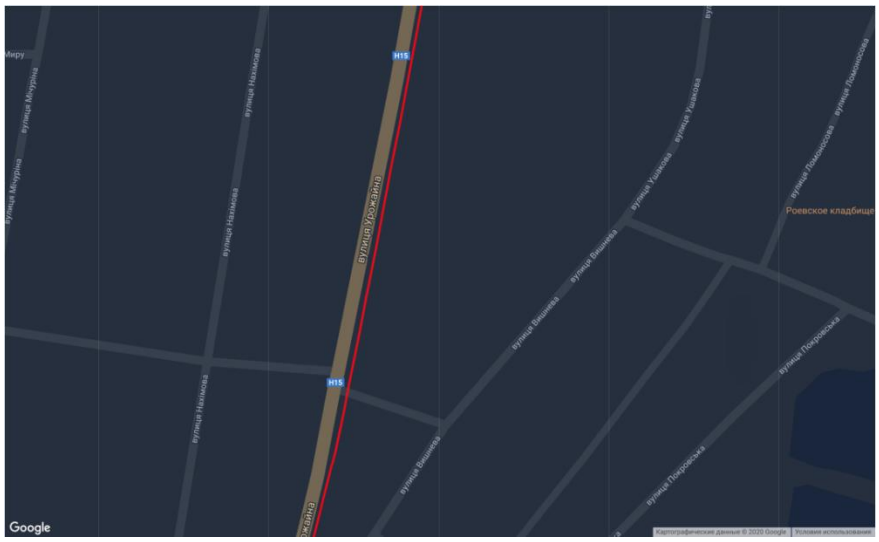


Рисунок 4.7 – Проїдений шлях 1 за даними GPS

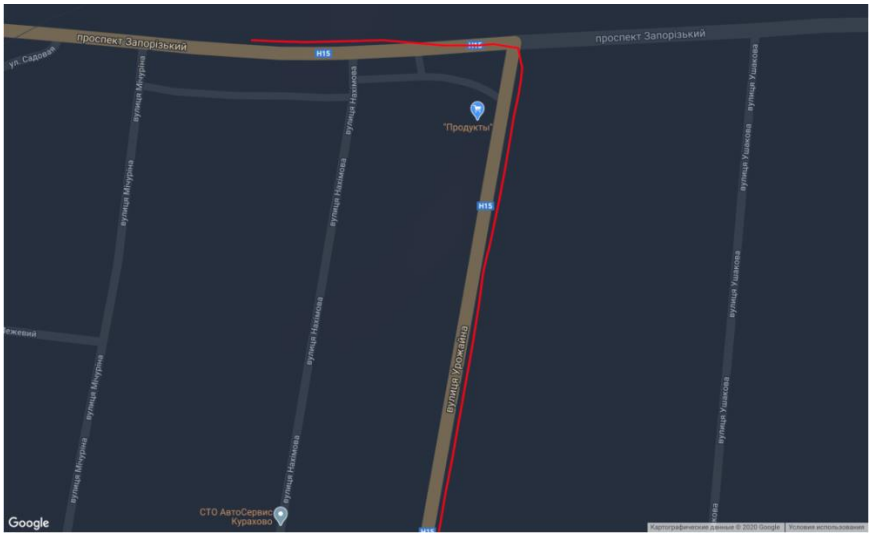


Рисунок 4.8 – Проїдений шлях 2 за даними GPS

Незважаючи на те, що автомобіль рухався рівно по трасі H15, на отриманих На рисунках 4.7 – 4.8 видно, що шлях містить невелику похибку, а саме шлях дещо зсунутий вправо. Через цю похибку при наступному аналізі буде важко групувати дані за місцевістю. Додаток орієнтовано на водіїв, тому користувач у більшості випадків буде знаходитись безпосередньо на дорозі, таким чином ми можемо самі апроксимувати поточне місцезнаходження користувача на найближчу дорогу. Але для цього потрібно мати доступ до цифрової версії карти місцевості. У додатку за основу було взято Google Maps, так як це одна з найпопулярніших, деталізованих та

легко налагоджуваних карт, також важливим фактором є те, що Google пропонує велику кількість допоміжних сервісів для роботи з картою[16]. Таким чином, перед збереженням зчитаних даних, місцезнаходження користувача закріплюється за найближчою дорогою. Подоланий шлях, після апроксимування, подано на рисунках 4.9 – 4.10.

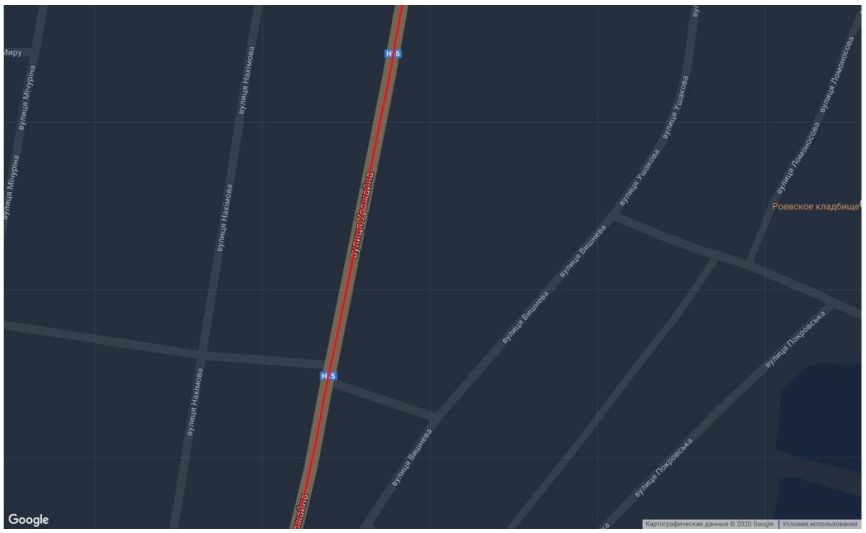


Рисунок 4.9 – Апроксимований подоланий шлях 1 за даними GPS

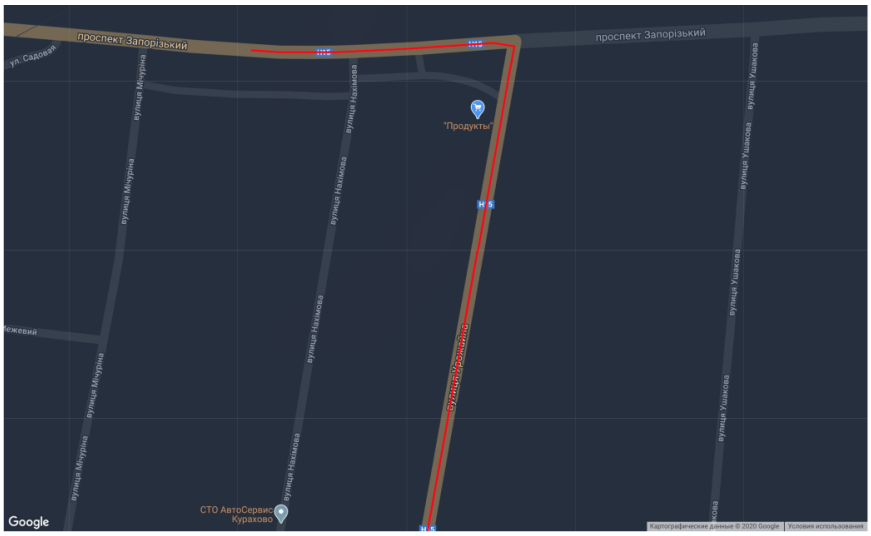


Рисунок 4.10 – Апроксимований подоланий шлях 2 за даними GPS

З апроксимованим подоланим шляхом можна більш зручно та детально працювати. Необроблена геолокація також зберігається, для можливого подальшого використання.

## 4.6. Маркування навчальних даних

Для початкового аналізу даних отриманих від сенсорів потрібно мати інформацію, яка зібрана з відміченими вручну мітками та поясненнями до цієї неї. Тобто система повинна мати можливість позначати кожну подію на дорозі під час збору даних, для забезпечення подальшого вибору ділянок, які містять важливу інформацію.

## 4.7. Автономність системи

Одна з найбільш істотних недоліків системи на базі смартфону є електроспоживання пристрою та його енергоємність. Тому було прийнято рішення оптимізувати додаток для його автономності. Для цього, усі можливі обчислювальні операції потрібно делегувати серверу.

Було проведено декілька експериментів з метою з'ясування найбільш енерговитратних процесів. Експерименти енергоспоживання пристрою було проведено на смартфоні iPhoneXS за допомогою інструментів xCode Energy Imapact та CPU Instruments. Експериментально було встановлено найбільш енерговитратні процеси:

- частота оновлення сенсорів руху
- обчислюванні операції для фільтрації та обробка даних перед відправкою на сервер
- точність GPS

Інтервал оновлення сенсорів руху розглядався у діапазоні 0.05 - 0.2, при більшій частоті оновлення було виявлено занадто велике значення енергоспоживання пристрою. Діаграми споживання електроенергії пристрою при інтервалах оновлення 0.05, 0.1, 0.2 подано на рисунках 4.11 – 4.13 відповідно.

За даними діаграм, які наведено вище, можна прослідкувати, що при інтервалі оновлень 0.05 енерговитрати дуже високі, але частота оновлень безпосередньо

впливає на якість отриманої інформації, тому потрібно зважати на отримані від сенсорів дані.

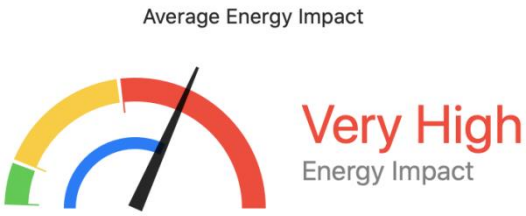


Рисунок 4.11 – Енергоспоживання при інтервалі оновлення 0.05

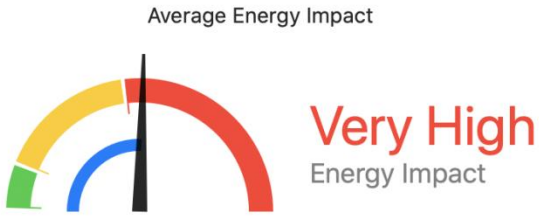


Рисунок 4.12 – Енергоспоживання при інтервалі оновлення 0.1

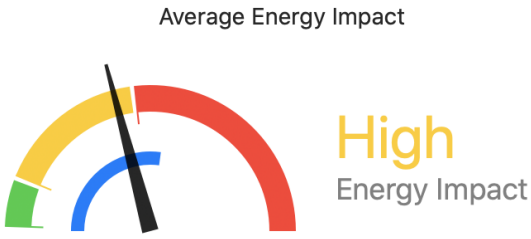


Рисунок 4.13 – Енергоспоживання при інтервалі оновлення 0.2

Дані, що отримані від сенсорів руху при інтервалах оновлень 0.05, 0.1, 0.2 подано на рисунках 4.14 – 4.16 відповідно.

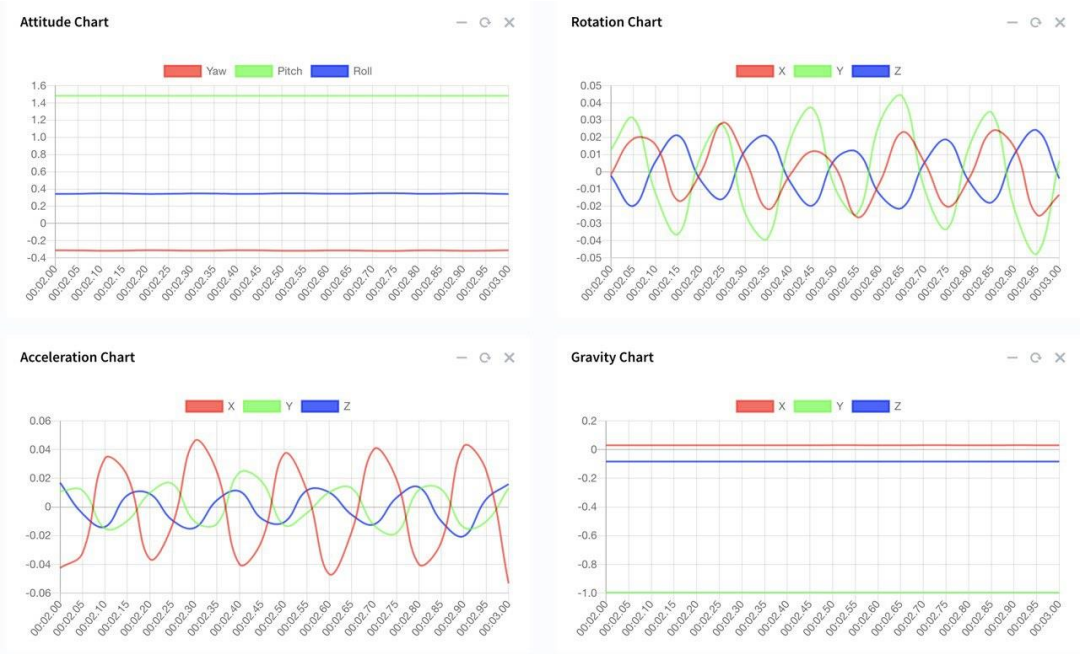


Рисунок 4.14 – Дані, отримані від сенсорів руху при інтервалі оновлень 0.05

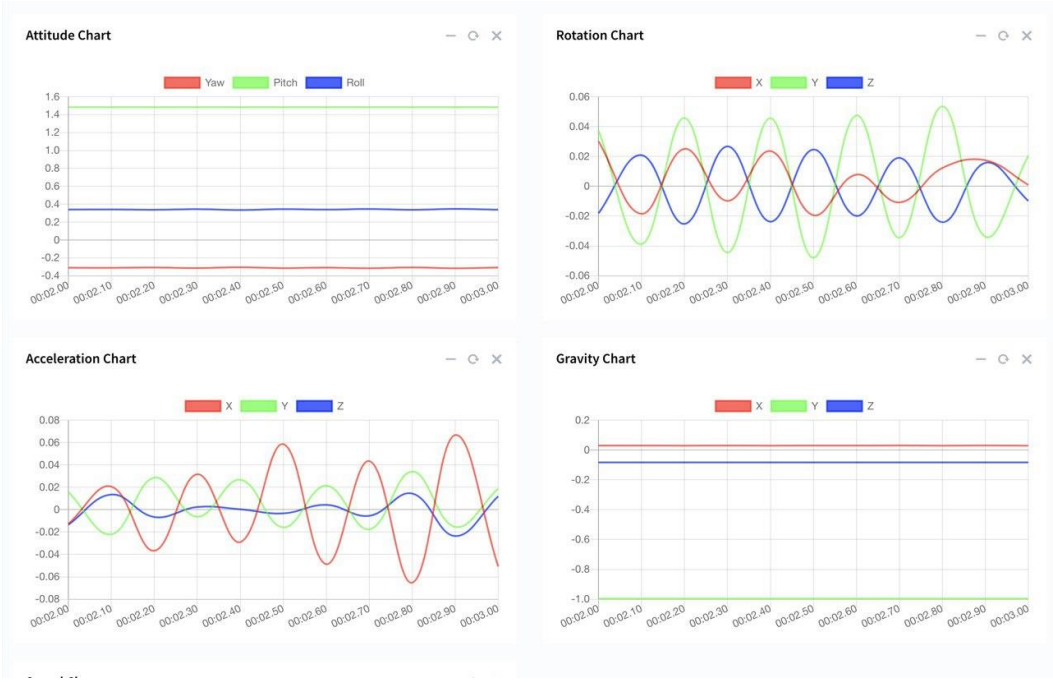


Рисунок 4.15 – Дані отримані від сенсорів руху при інтервалі оновлень 0.1

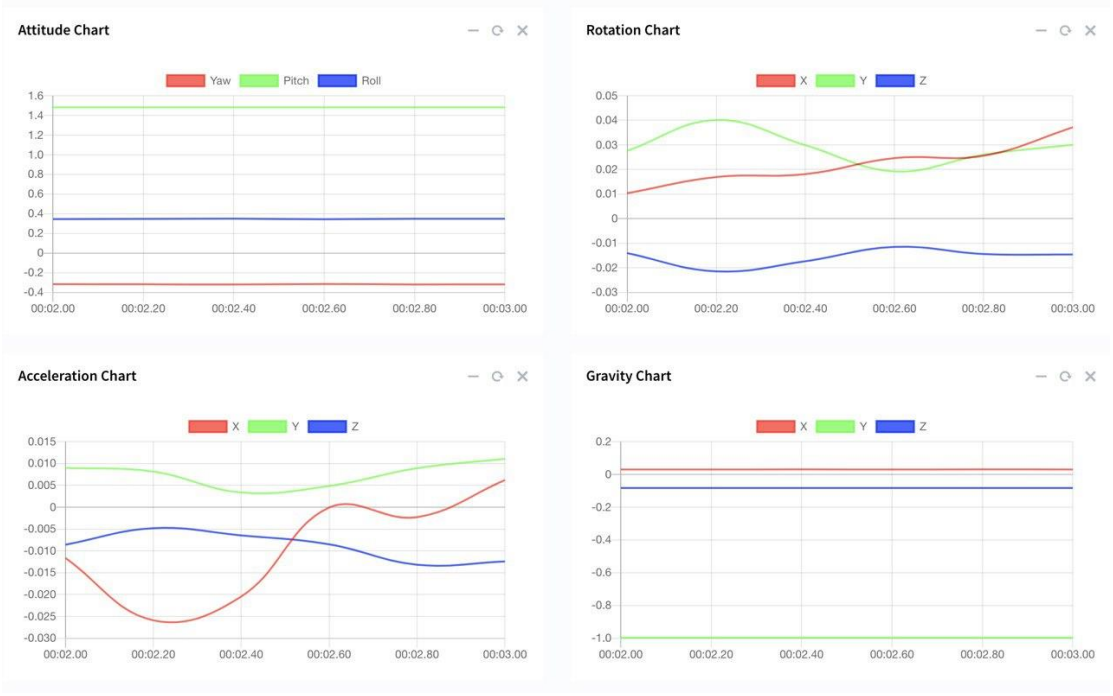


Рисунок 4.16 – Дані, отримані від сенсорів руху при інтервалі оновлень 0.2

За отриманими графіками можна побачити, що данні отримані при значеннях інтервалів оновлення 0.05 та 0.1 дуже схожі. Щодо інтервалу оновлень 0.2, можна

прослідкувати, що дана частота не може зафіксувати потрібну точність зміни даних, тому далі вона розглядатися розглядатись не буде.

При подальшому розгляді частоти оновлень сенсорів руху було обрано частоту 0.1. Причиною тому, стало те що при частоті оновлень 0.1, споживається не така значна кількість енергії як при частоті оновлень 0.05, а результуючі значення даних сенсорів у обох випадках мають дуже незначні розбіжності.

Зменшити енерговитрати ще на 30% вдалося за рахунок перенесення усіх можливі обчислюваних операцій на сервер.

Фінальне значення енерговитрат мобільного додатку подано на рисунку 4.17.

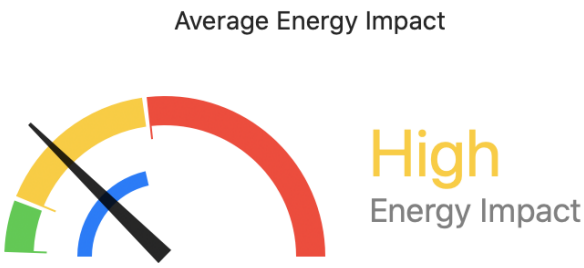


Рисунок 4.17 – Енергоспоживання після переносу обчислюваних операцій на сервер

Також була можливість зменшити енергоспоживання додатку за рахунок зменшення точності роботи GPS, але зважаючи на те, що це одна з найважливіших метрик у системі, точність її роботи було залишено найвищою з можливих.

В результаті вдалося отримати значно менший рівень енергоспоживання додатку, ніж було спочатку, але він залишається достатньо великим, за рахунок відображення мапи, бо цей процес потребує багато ресурсів.

## 4.8. Алгоритм виявлення аномалій

Алгоритм програмної системи, яка розроблюється, починається з ручного збору тренувальних даних. Початкова інформація ,яка ретельно і багаторазово збирається з сенсорів, використовується для визначення розташування різних класів

дорожніх аномалій та іншої корисної інформації про дорожнє покриття на певних ділянках.

З метою виявлення аномалій дорожнього покриття за допомогою порогових підходів було проаналізовано зміну значень показників сенсорів, та отримано стандартні відхилення, на основі цих даних. Було проведено контроль амплітуди сигналу акселерометра та виявлено паттерни аномалій в сигналі.

Шаблони аномалій в цифрових сигналах виникають, коли потужність сигналу перевищує певне значення. За основу, для розрахунку порогових значень, були розглянуті три питання: тривалість інтервалу для функції вікна, фіксованого проти гнучкого визначення порогу та амплітуда сигналу відносно інших властивостей амплітуди сигналу (середнє та стандартне відхилення).

Визначення довжини інтервалу для функції вікна в спектральному аналізі є складним завданням, оскільки воно пов'язане з різними факторами, такими як швидкість руху транспортних засобів та відстань від переднього до заднього колеса. Функція вікна враховує попередньо визначені інтервали сигнальних даних для аналізу та вилучення функцій на відміну від перегляду сигнальних даних окремо.

Визначення правильних порогових значень у статистичному підході є інтенсивним процесом, оскільки на значення даних дорожньої аномалії впливають різні умови. Система підвіски автомобіля, сенсорні властивості смартфонів та розміщення смартфонів все впливають на те, як смартфони відчувають одну й ту саму аномалію.

Як зазначено вище, ми використовуємо триосевий акселерометр для нашого аналізу. Акселерометр має тривимірну декартову систему координат відносно себе, представлену ортогональними осями x, y, z. Крім того, ми визначаємо декартову систему відліку відносно транспортного засобу, в якому знаходиться акселерометр.

Проблема виявлення аномалій дорожнього покриття є складною з кількох причин. Основною задачею є встановлення правильного та істинного порогового значення для наступного аналізу. Зазвичай це робиться лише за допомогою анотації вручну, яка є суб'єктивною і може значно відрізнятися в залежності від того, як

транспортний засіб наближається до аномалії та з якою швидкістю. Однак амплітуда сигналу може мати різні характеристики з різною швидкістю. Це можна побачити на рисунках 4.18 і 4.19, де продемонстровано, сигнали датчиків автомобіля, що пересувається по поганій дорозі на низькій та високій швидкостях.

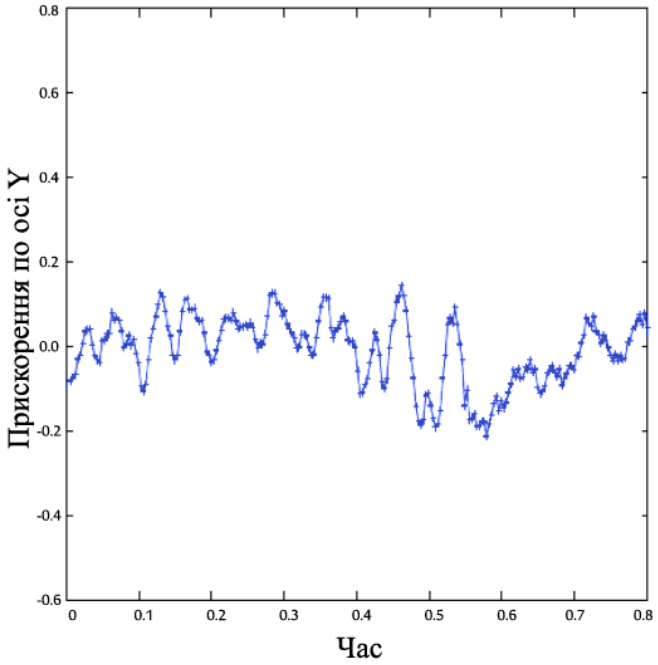


Рисунок 4.18 – Значення осі y відносно автомобіля на поганій дорозі при низькій швидкості

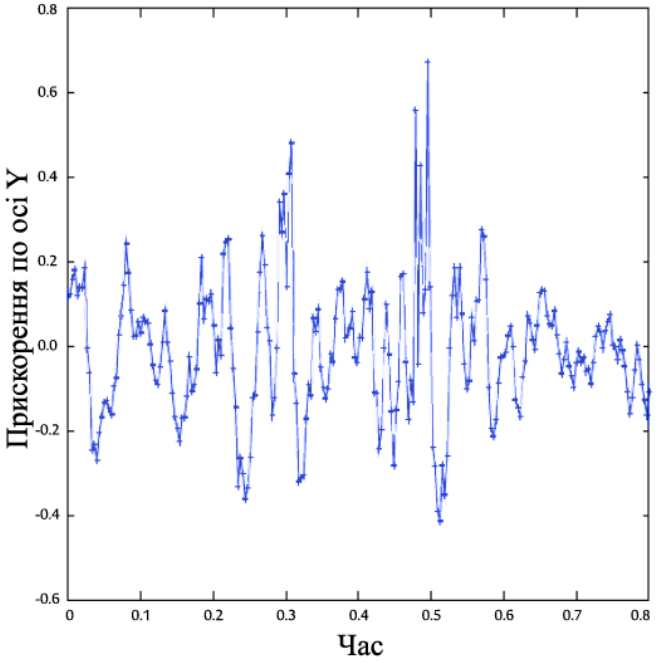


Рисунок 4.19 – Значення осі y відносно автомобіля на поганій дорозі при високій швидкості

Зауважимо, що при низьких швидкостях помітне занурення значень нижче 0.1, що витримується у кількох випадках. Ми припускаємо, що це занурення відповідає фізичному явищу з моменту, коли колесо автомобіля переходить через вибоїну до тих пір, поки колесо не зустрінеться з дном вибоїни.

Коли колесо ударяється об землю, на транспортний засіб передається різка сила, яка зареєстрована як точка додатного локального екстремуму на графіку. На низьких швидкостях значення висхідної точки локального екстремуму є менш вираженим, як на рисунку 4.18, тоді як на високих швидкостях її значення може бути значним, як на рисунку 4.19.

Було виявлено, що на великій швидкості було набагато більше помилкових спрацьовувань. Припущено, що невеликі хвилі на дорозі можуть бути причиною таких помилково-позитивних підйомів.

В результаті вищезазначених спостережень, було прийнято рішення про використання двох алгоритмів для класифікації ударів залежно від швидкості руху транспортного засобу.

На високій швидкості (> 30 км / год) використано різке зростання сигналу для виявлення ударів, де точка додатного локального екстремуму класифікується як підозрілий удар.

На низьких швидкостях пропонується новий алгоритм класифікації удару, який шукає тривале занурення значень датчику, яке досягає значень нижнього порогу. Нарешті, для того, щоб визначити швидкість руху автомобіля, ми можемо покластися на локалізацію GPS, щоб дізнатися рухається транспортний засіб з низькою швидкістю (<30 км / год) або ні.

Також було проведено декілька випробувань на ділянках з поганим та добрим дорожнім покриттям довжиною приблизно 10-15 км.

У випадку коли швидкість низька, поріг T становить 0.2, його було отримано після проведення експерименту на поганій ділянці дороги, а потім той самий поріг було використано для більш тривалого відрізку дороги для перевірки його коректності.

У випадку коли швидкість автомобіля висока, можна використовувати методи динамічного налаштування порогу на різні швидкості. Однак у даній роботі проілюстровано кращий сценарій. Налаштовано пороговий параметр окремо на його оптимальні значення для низьких та високих швидкостей відповідно. У випадку коли швидкість висока, поріг T становить 0.42, але цей поріг дає багато хибних показників при дуже високих швидкостях.

На рисунку 4.20 подано блок-схему алгоритму пошуку потрібного розміру вікна. На рисунок 4.21 проілюстровано блок-схему алгоритму розпізнавання аномалій дорожнього покриття.

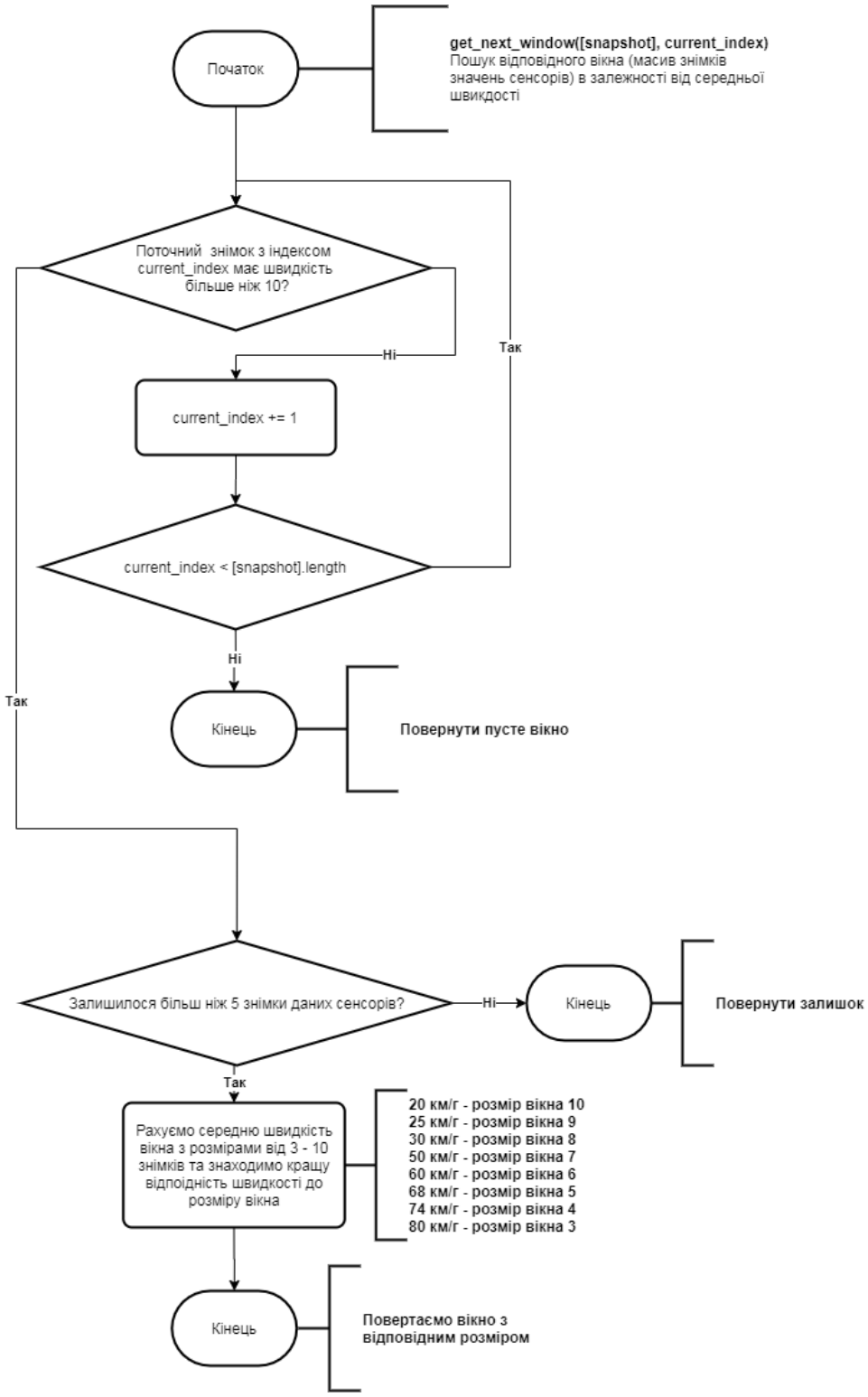


Рисунок 4.20 – Блок-схема алгоритму пошуку потрібного розміру вікна.

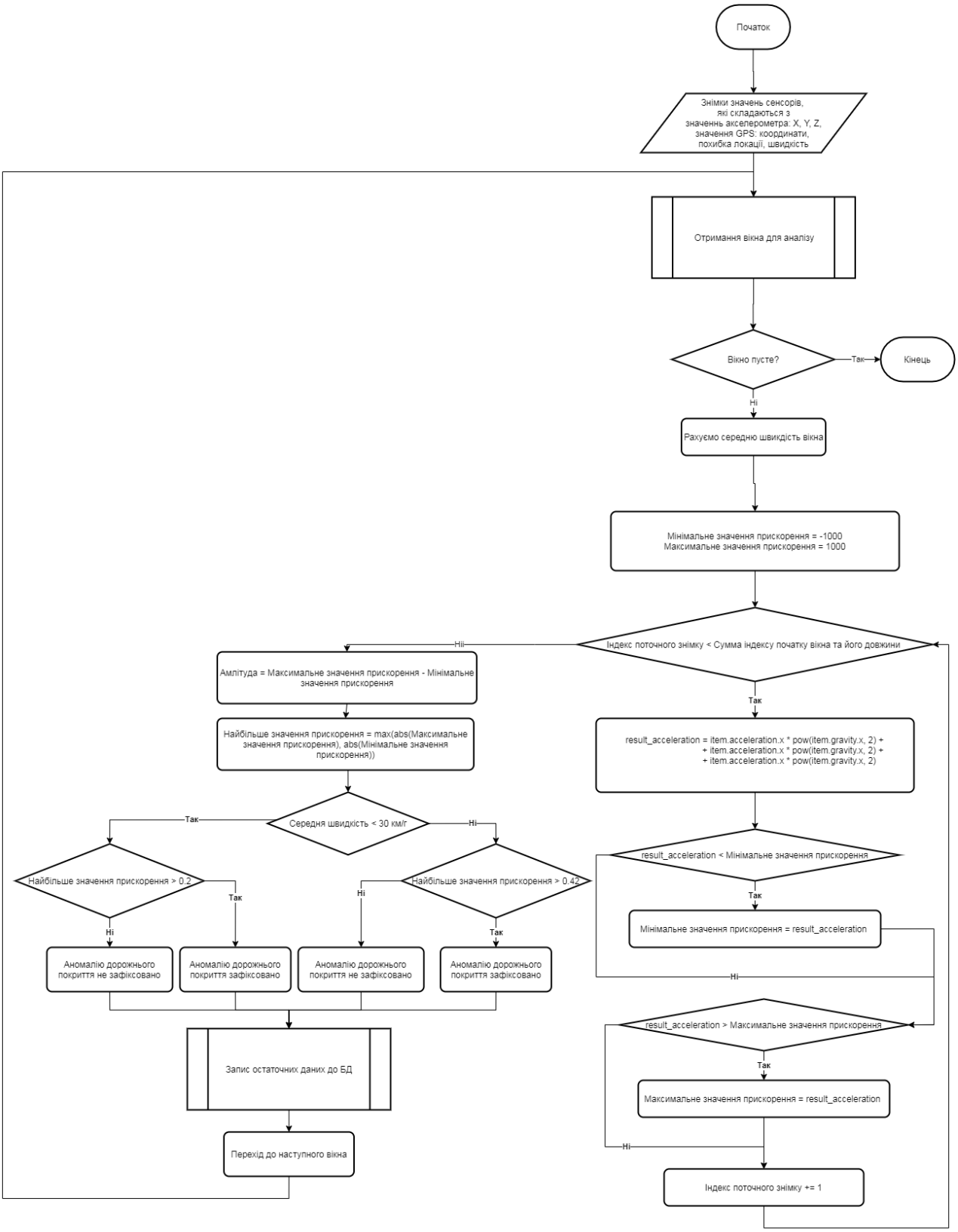


Рисунок 4.21 – Блок-схема алгоритму розпізнавання аномалій дорожнього покриття.

# 5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

## 5.1. Мобільний додаток користувача

При вході до мобільного додатку, додаток запитує дозвіл на використання геопозиції користувача. У випадку, коли користувач відмовиться, додаток не буде коректно працювати. Якщо користувач дозволить використання геоданих, то його зустрічає географічна мапа, яка демонструє його поточне місцезнаходження. Візуалізацію подано на рисунках 5.1 – 5.2.

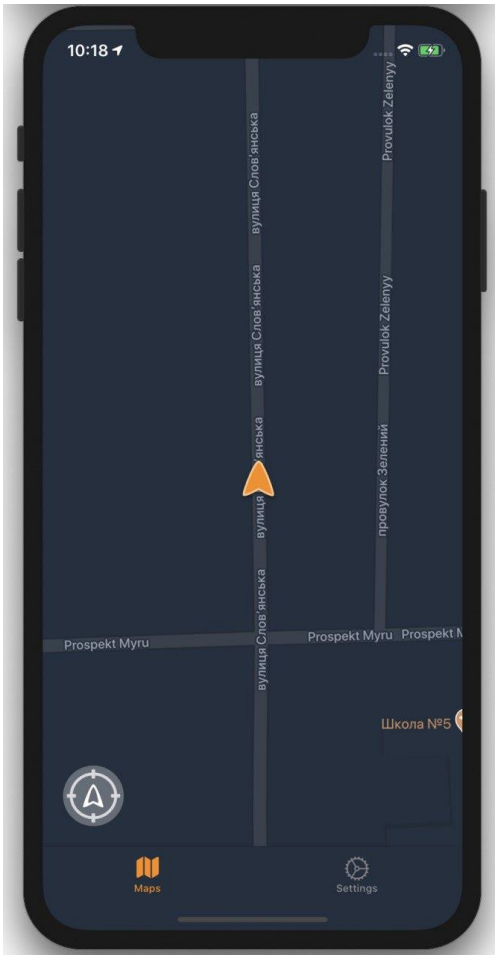


Рисунок 5.1 – Мапа у неактивному стані

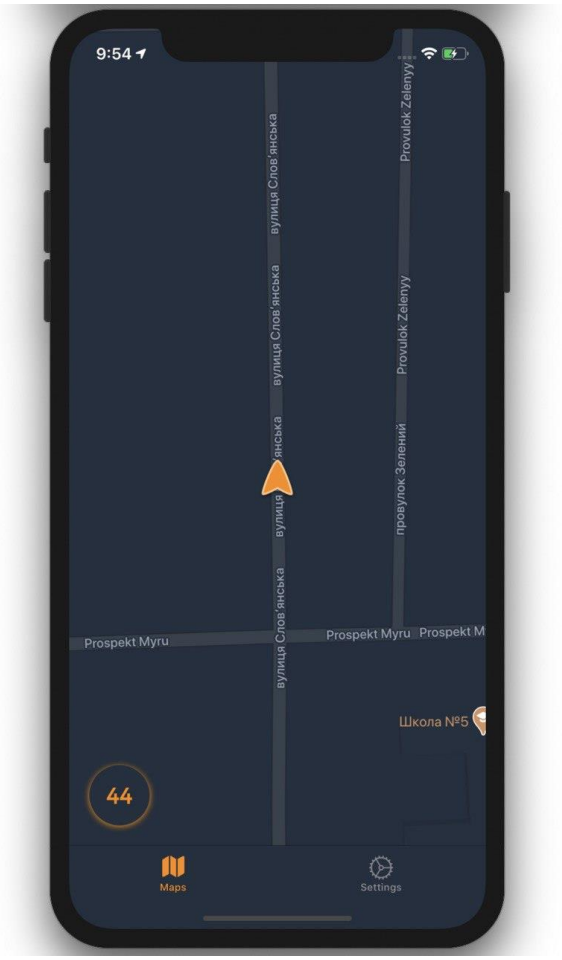


Рисунок 5.2 – Мапа у стані подорожі

У нижньому кутку екрану можна побачити кнопку "Закріпити камеру за користувачем", після натискання на неї, додаток переходить у стан подорожі,

кнопка навігації замінюється спідометром(Рисунок 5.2), слідую за користувачем на карті. У цьому стані додаток паралельно починає слідкувати за даними сенсорів і якщо вони проходять фільтрацію, то відправляються на сервер, для подальшого аналізу. Якщо користувач проведе (свайпне) по карті додаток перейде у неактивний стан (Рисунок 5.1).

Також користувач може побачити стан деяких відрізків дороги, де вже було зібрано достатньо даних для присвоєння статусу (Рисунок 5.3). В залежності від стану дороги, стежку, на екрані, буде виділено відповідним кольором (зелений – дорога в гарному стані, жовтий – дорога нормальна, помаранчевий – стан дороги поганий, красний – стан дороги дуже поганий)

Друга навігаційна вкладка є Налаштування, після натискання на неї, користувач потрапляє на список усіх можливі налаштування додатку (Рисунок 5.4).

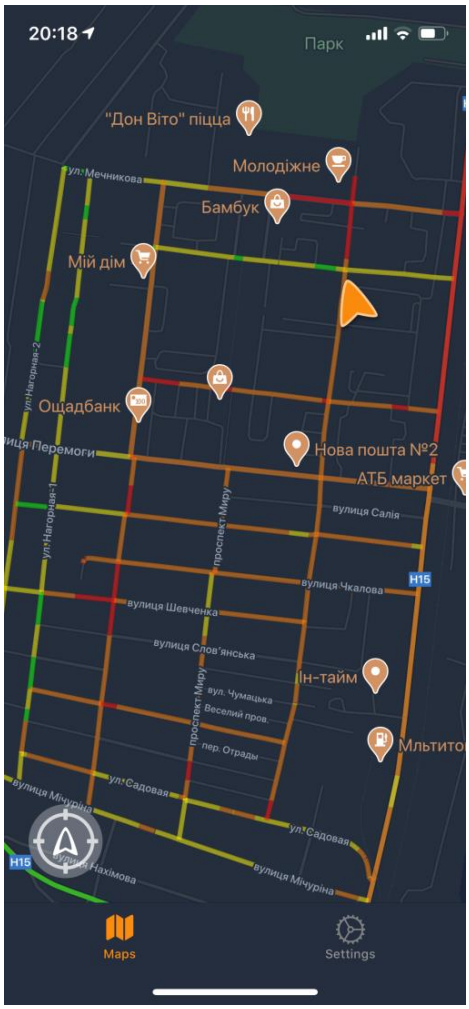


Рисунок 5.3 – Стан дороги

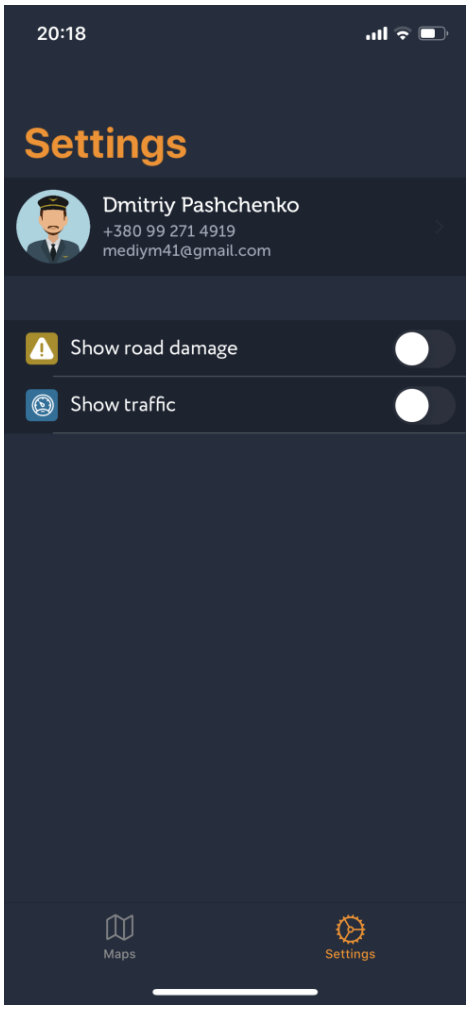


Рисунок 5.4 – Налаштування у користувача

Користувач має дві опції для перемикання: відображати стан дороги, відображати завантаженість доріг. Якщо перемикач увімкнено, то відповідні елементи будуть відображатись карті (на першій навігацій вкладці Карти).

Коли перемикач Відображати увімкнено, то на вкладці Карти можна побачити завантаженість деяких відрізків дороги, де вже було зібрано достатньо даних для присвоєння статусу(Рисунок 5.5). В залежності від завантаженості дороги, стежку, на екрані, буде виділено відповідним кольором (зелений – дорога вільна, жовтий – дорога мало завантажена, помаранчевий – дорога середнє завантажена, красний – дорога сильно завантажена)

Також у додатку, як у всіх сучасних навігаторах, є можливість прокласти маршрут до певного місця. Для цього користувачу потрібно здійснити довге натискання на точці на карті, в яку він хочу потрапити. Додаток поставить в цій точці мітку, прокладе до неї маршрут, покаже інформацію про побудований маршрут (приблизна тривалість подорожі, дистанція до точки призначення, та приблизний стан дороги через який він пролягає)(Рисунок 5.6).

Щоб розпочати подорож, користувачу потрібно натиснути кнопку Start(Старт), та у протилежному випадку, кнопку Cancel(Відмінити).

Якщо користувач натисне кнопку Start(Старт), то додаток перейде у режимі та прибере модальне вікно з інформацією про маршрут(Рисунок 5.7).

Якщо користувач натисне кнопку Cancel(Відмінити), то додаток прибере усі графічні елементи, відносно маршруту. Щоб прибрати маршрут, користувачу потрібно натиснути на маркер, котрий встановився після прокладки маршруту, після чого з'явиться модальне вікно, у котрому користувачу потрібно підтвердити своє рішення, натиснувши кнопку Stop(Стоп)(Рисунок 5.8).

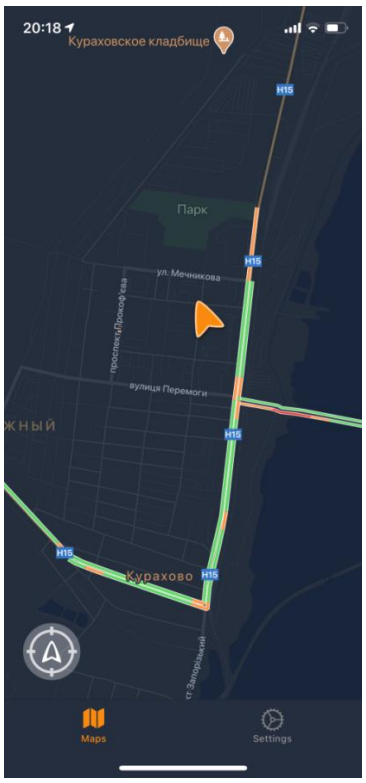


Рисунок 5.5 – Завантаженість на дорозі

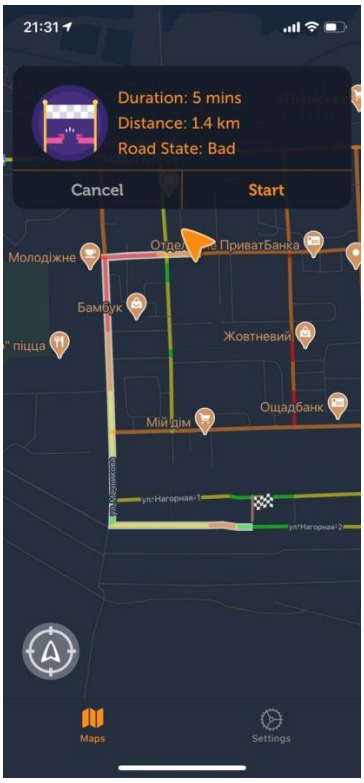


Рисунок 5.6 – Прокладення маршруту та початок подорожі

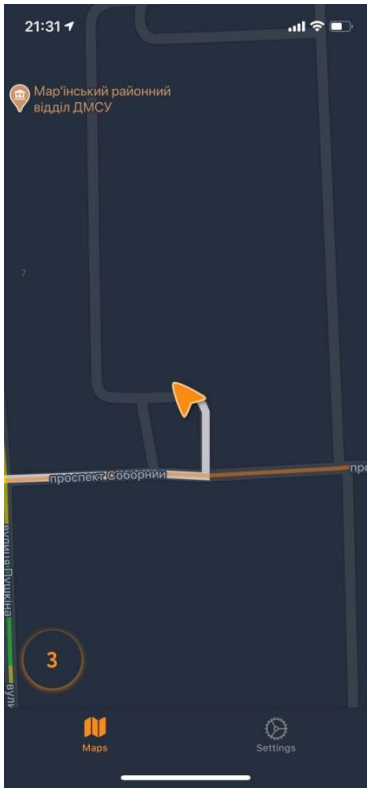


Рисунок 5.7 – Режим подорожі після прокладання маршруту

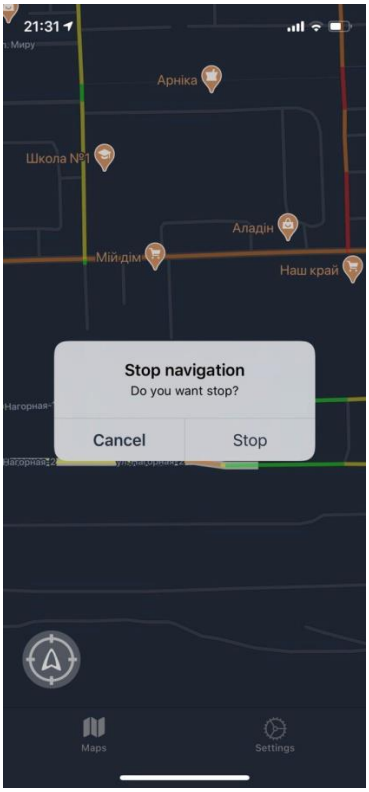


Рисунок 5.8 – Модальне вікно для скасування прокладеного маршруту

## 5.2. Розширена версія мобільного додатку

Розширена версія має такий самий інтерфейс та схему взаємодії, як і в базовому додатку. Проте, в ньому присутні декілька додаткових вкладок (Sensors(Сенсори) та Record(Запис)). ЇЇ подано на рисунку 5.9.

На рисунку 5.10 проілюстровано налаштування у адміністратора.

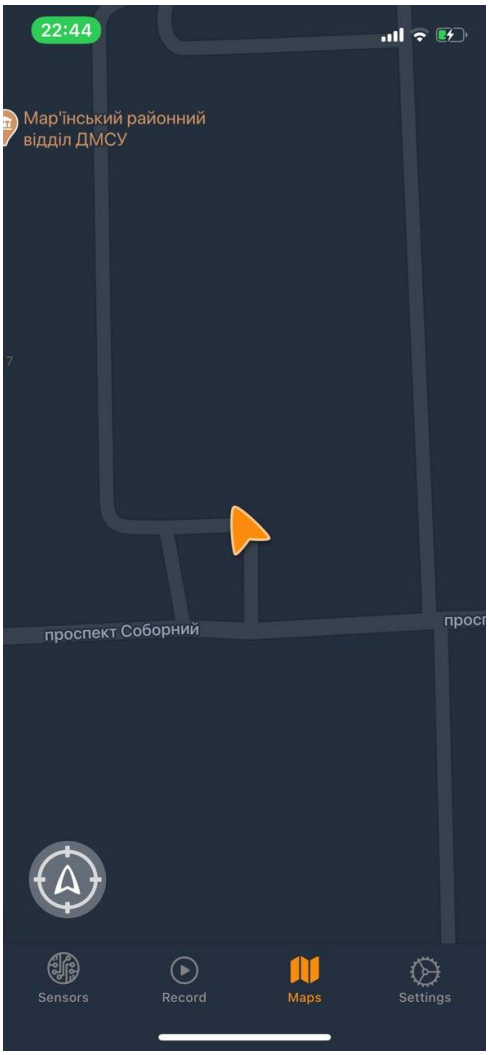


Рисунок 5.9 – Розширена версія додатку

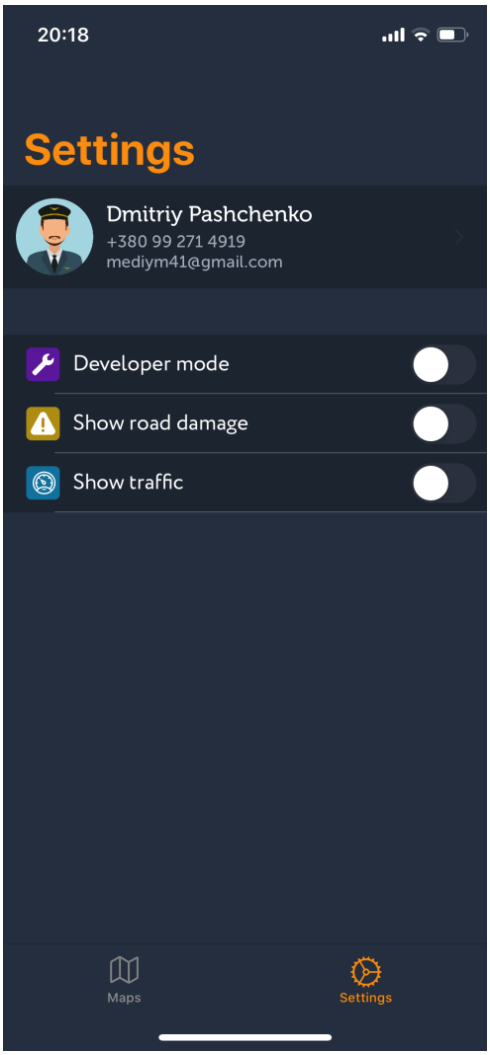


Рисунок 5.10 – Налаштування у адміністратора

Вкладка Sensors(Сенсори) дозволяє користувачу додатку переглядати значення показників, отриманих від сенсорів(Рисунки 6.11-6.14). Тим самим імітувати певні патерни поведінки.

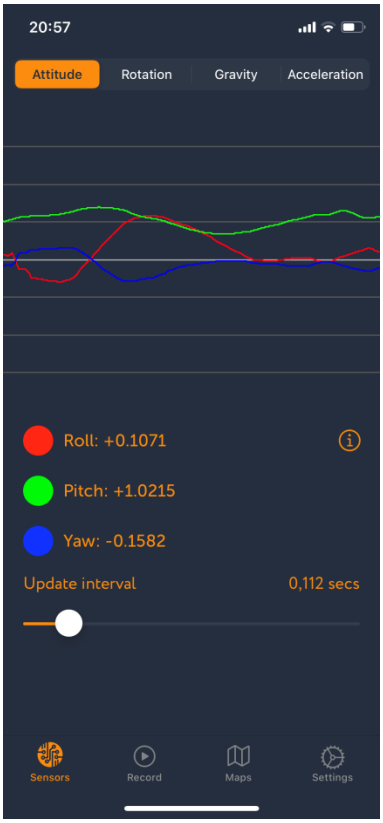


Рисунок 5.11 – Значення Attitude

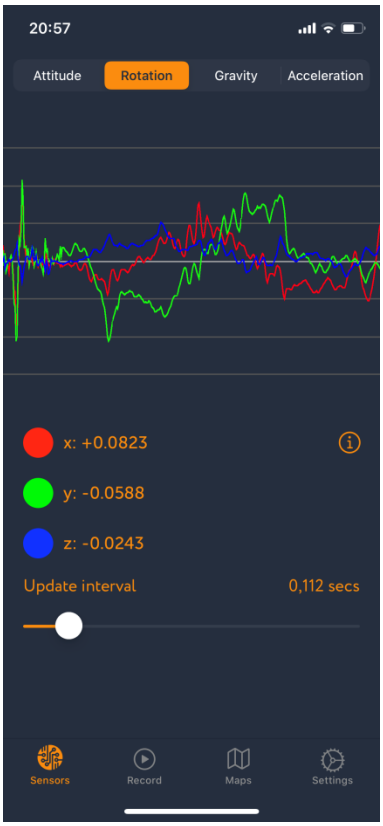


Рисунок 5.12 – Значення Rotation

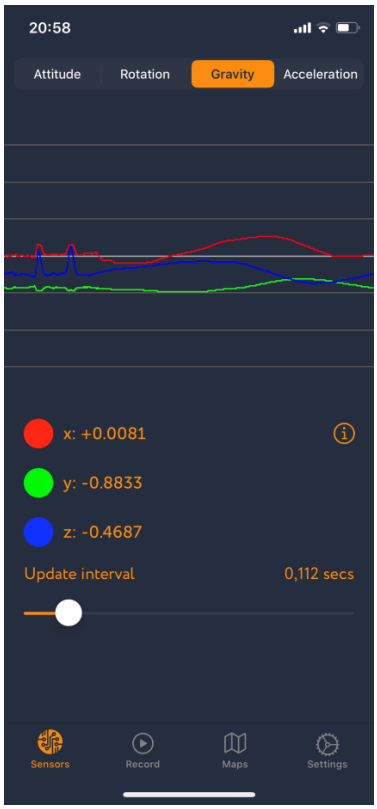


Рисунок 5.13 – Значення Gravity

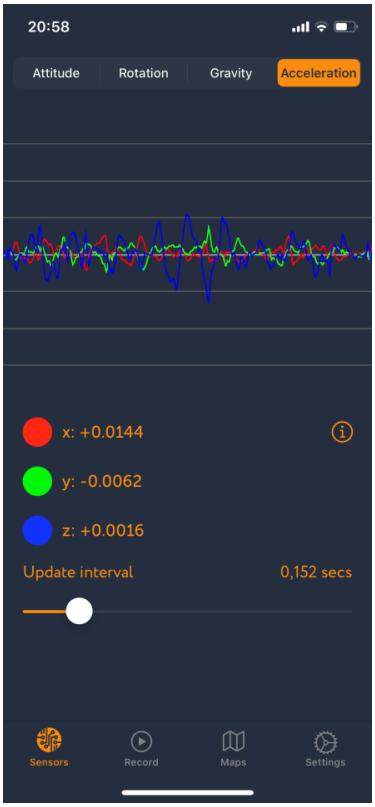


Рисунок 5.14 – Значення Acceleration

Для полегшення розуміння розміщення векторів(x, y, z), було додано вкладку Info(Інформація), на котрій можна переглянути візуально інтерпретаціювекторів(Рисунки 5.15-5.16).

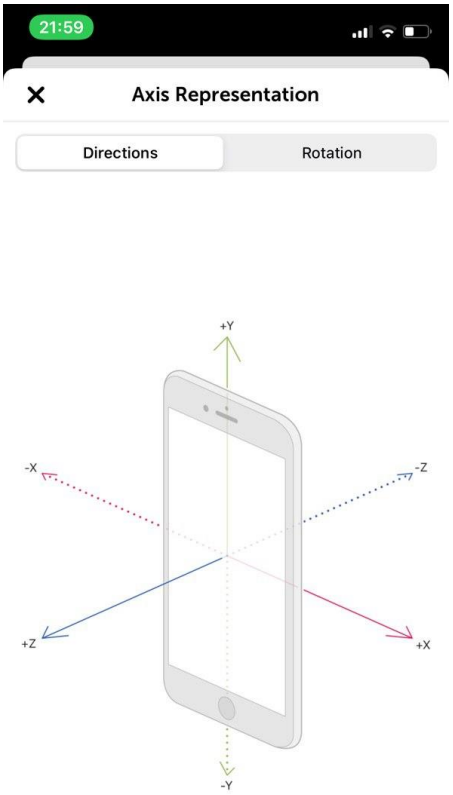


Рисунок 5.15 – Візуальна інтерпретація напрямків осей

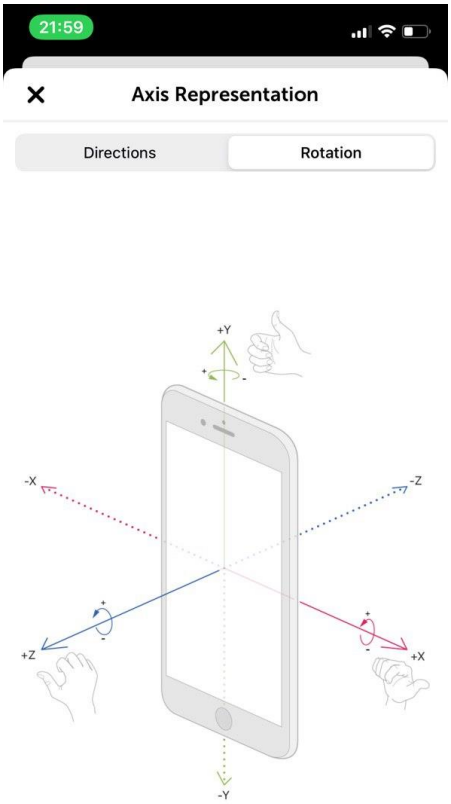


Рисунок 5.16 – Візуальна інтерпретація напрямків обертання

Вкладка Records довзоляє користувачу додатку записувати конкретні ділянки, та спостерігати за важливими даними у режимі реального часу. Усе що потрібно це вибрати певну частоту оновленя даних та натиснути кнопку Record(Запис), в наслідок чого, екран заповниться поточними даними.

Для того, щоб зупинити запис потрібно нажати кнопку Stop(Стоп), після чого додаток покаже модальний діалог у якому потрібно ввести імя файлу, до якого буде записано дані (Рисунки 5.17 – 5.19).

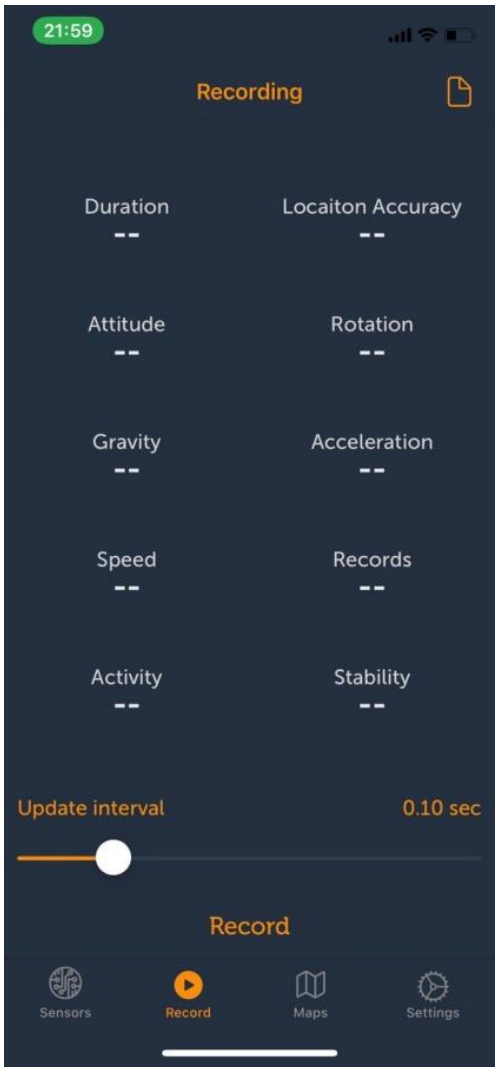


Рисунок 5.17 – Неактивний стан

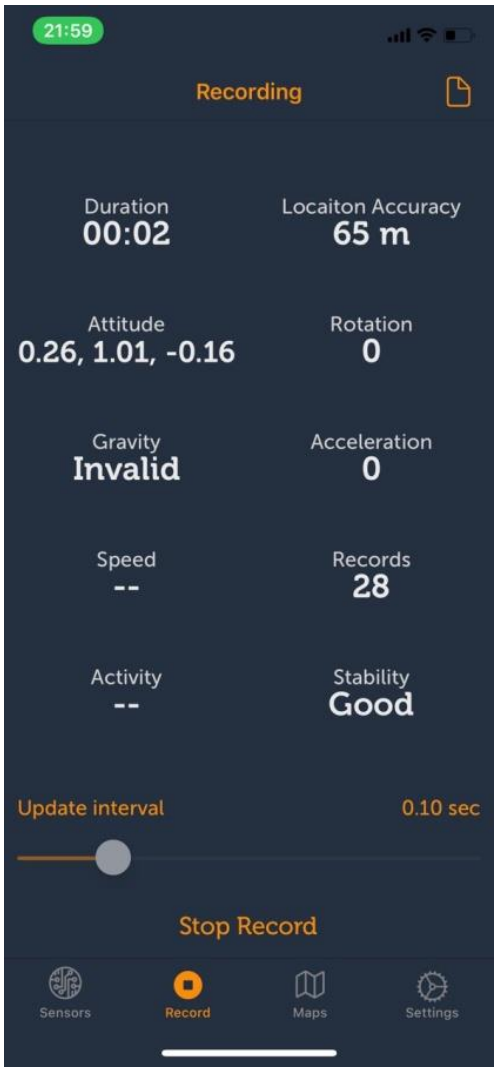


Рисунок 5.18 – Активний стан

Після цього файл збережеться локально, щоб його кудись відправити потрібно натиснутикнопку файл, що розташована у верхньому правому куті екрану. З'явиться список усіх записів, у якому буде відображено, у тому числі й тільки що створений файл. Його потрібно обрати та вибрати місце призначення відправлення. Щоб видалити записи потрібно нажати кнопку Clear(Очистити)(Рисунок 5.20).

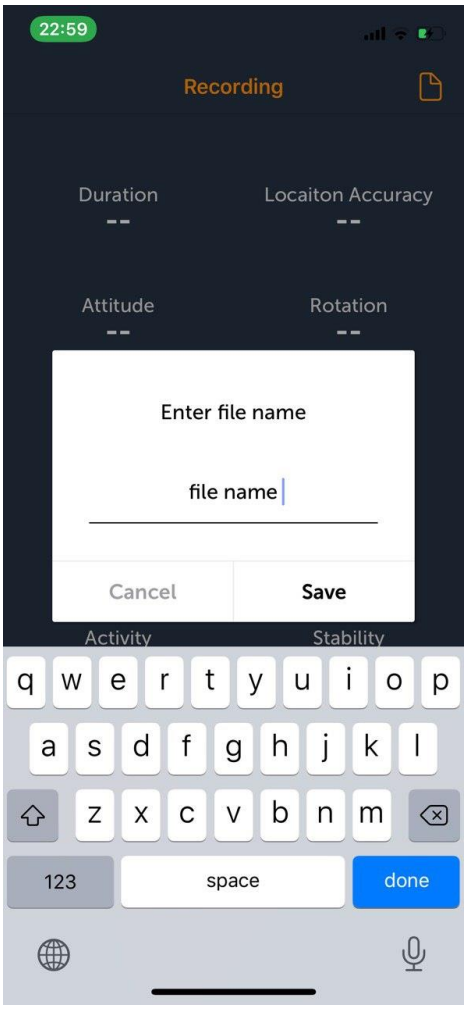


Рисунок 5.19 – Збереження запису

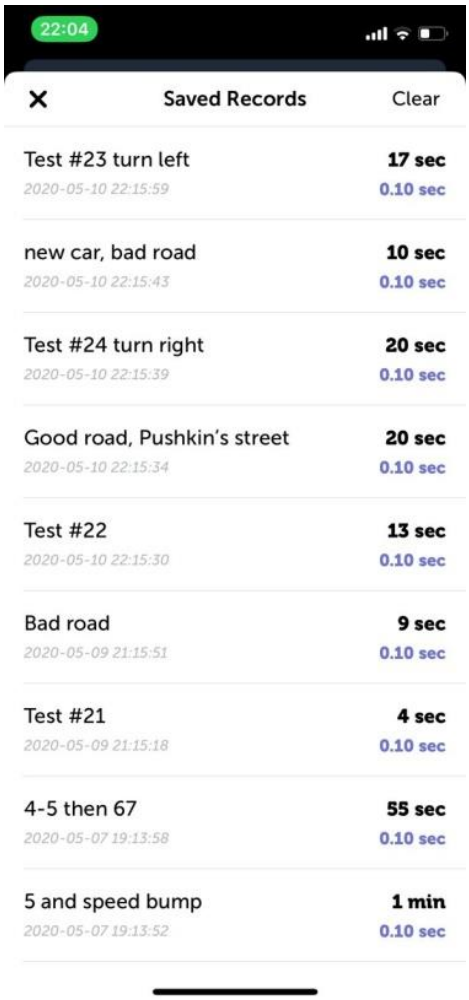


Рисунок 5.20 – Список записів які зберігаються локально

## 5.3. Веб-додаток

Веб-додаток призначений для перегляду даних, в зручному для аналізу вигляду. Для початку роботи з даними, користувачу потрібно пройти процедуру авторизації, а саме ввести електронну пошту та пароль. Після натискання кнопки Sign In (Авторизуватися), користувач може розпочати роботу з додатком.

Форму авторизації, наведено на рисунку 5.21.

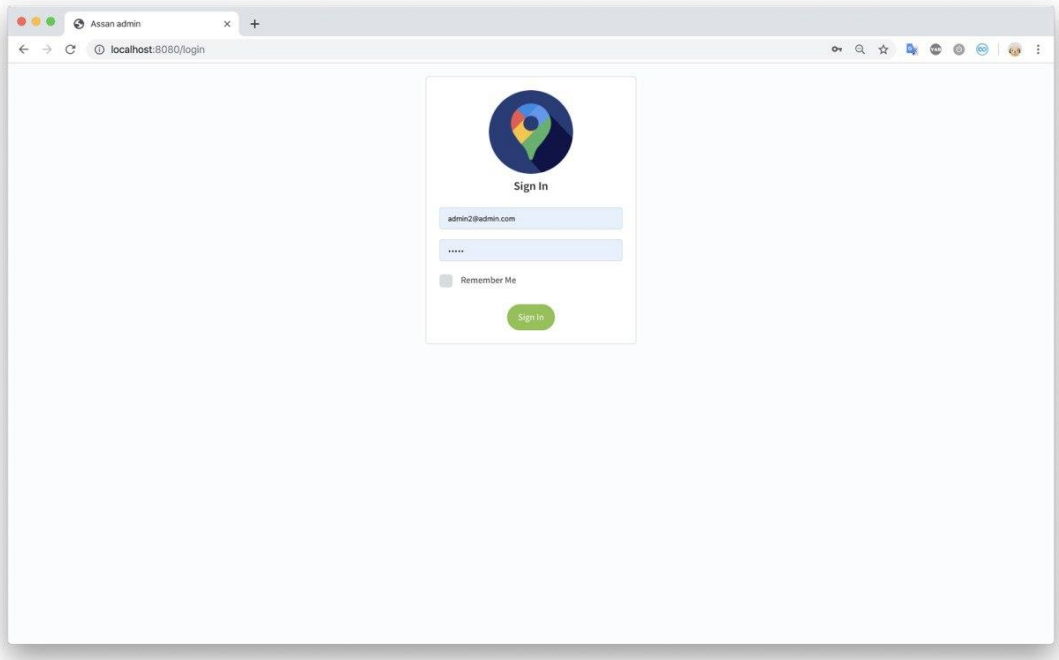


Рисунок 5.21 – Форма авторизації.

На рисунку 5.22 Наведено вкладку Dashboard(Панель приладів). Вона показує загальний стан системи.

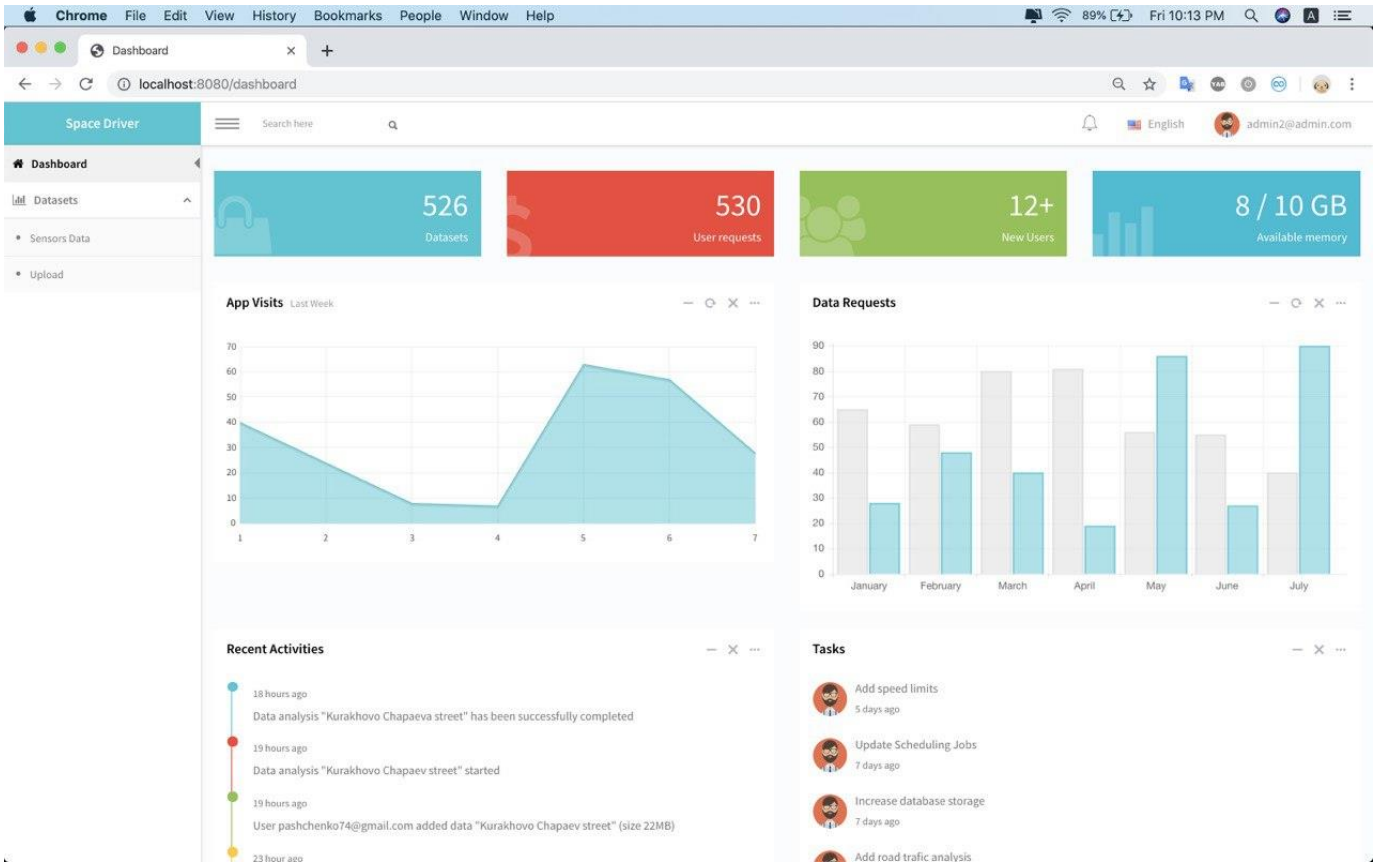


Рисунок 5.22 – Dashboard.

На рисунку 5.23 наведено вкладку Datasets (Набори даних). На вкладці користувач може додати файл з даними. На рисунку 5.23 наведено ситуацію, коли ще не додано жодного файлу.

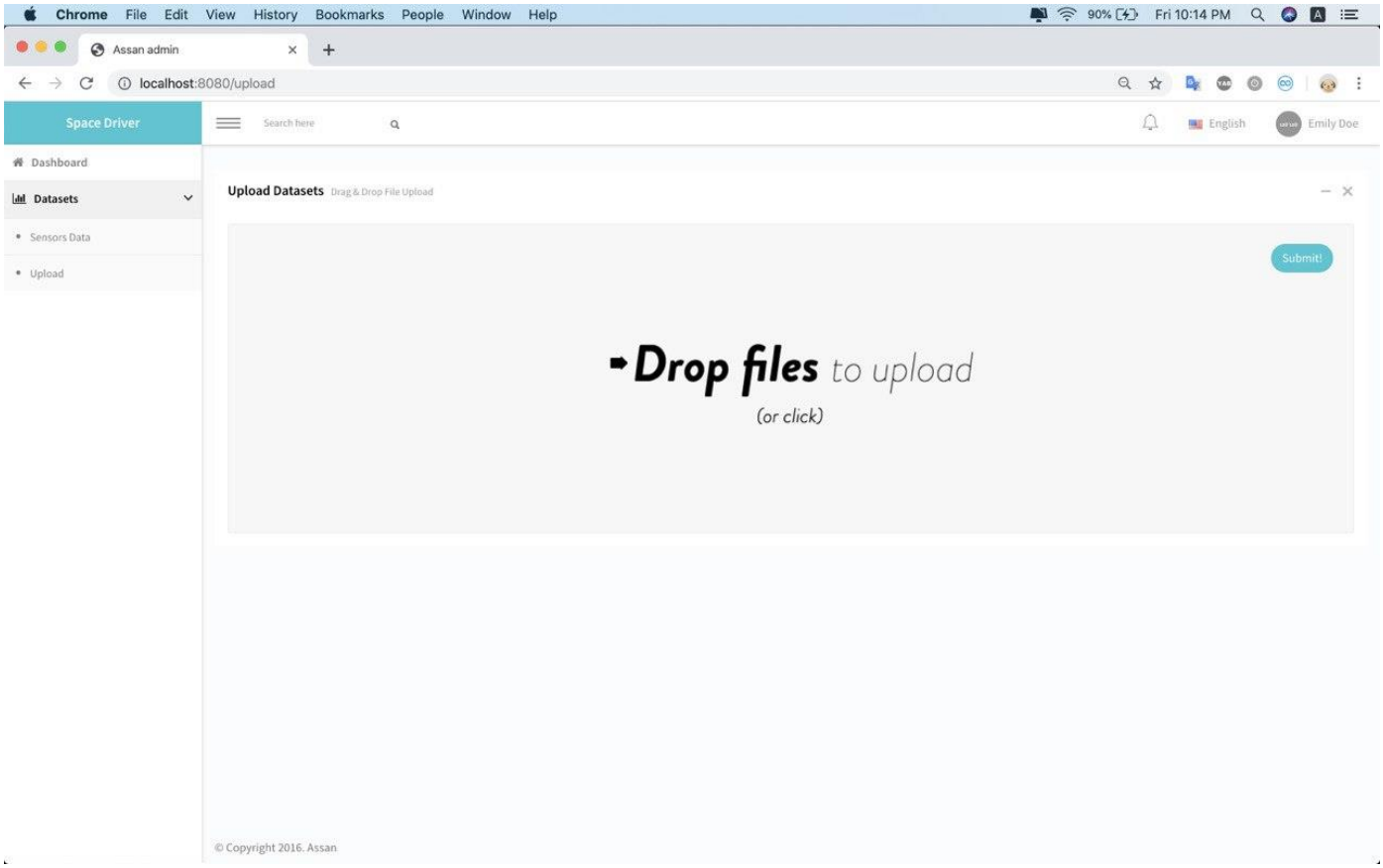


Рисунок 5.23 – Datasets.

На рисунку 5.24 приведено вкладку Datasets(Набори даних), на якій можна обрати таблиці даних. Інтерфейс має можливість сортування та фільтрації. Натиснувши на будь-який заголовок стовпця, додаток відсортує вибірку за зростанням по відповідному стовпцю. Якщо повторити цю операцію, додаток відсортує по тим же критеріям, але вже за спаданням. Щоб знайти запис с певною назвою, потрібно ввести текст, що потребується, після чого додаток прибере усі записи, котрі не містять введений текст. Якщо записів буде дуже багато, то користувач може скористатись пагінацією або розширити кількість елементів у списку.

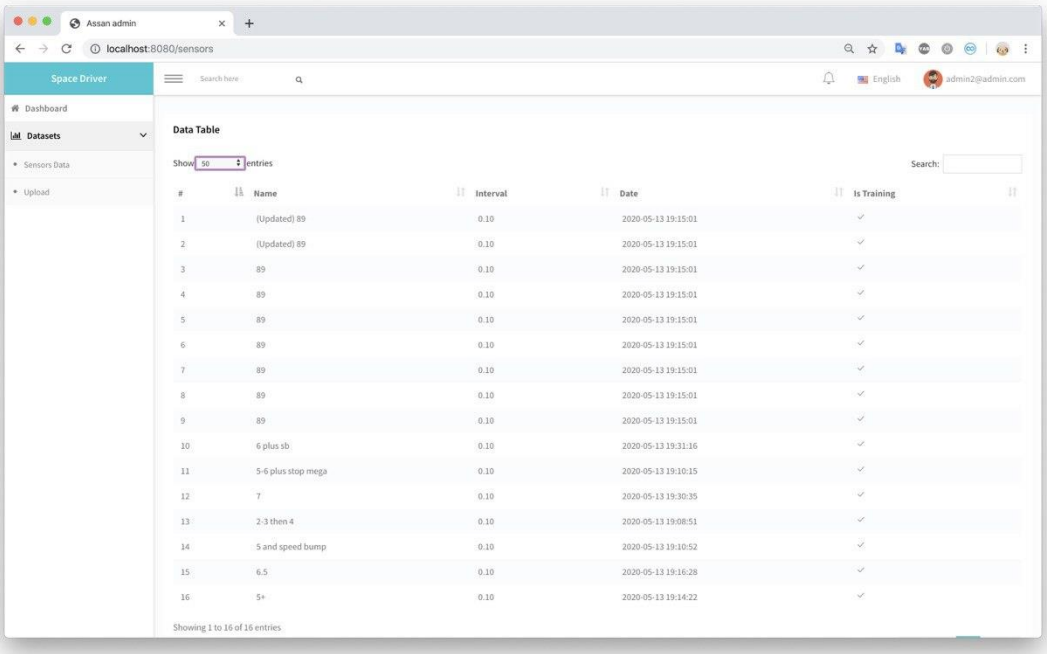


Рисунок 5.24 – Datasets (Data Table).

На вкладці Upload можна завантажити зібрані дані вручну. Для цього потрібно перетягнути файл у відповіну зону та відпустити або нажати кнопку Submit та выбрати потрібний файл у модальному вікні. На рисунку 5.25 показано випадок, коли дані вже завантажено.

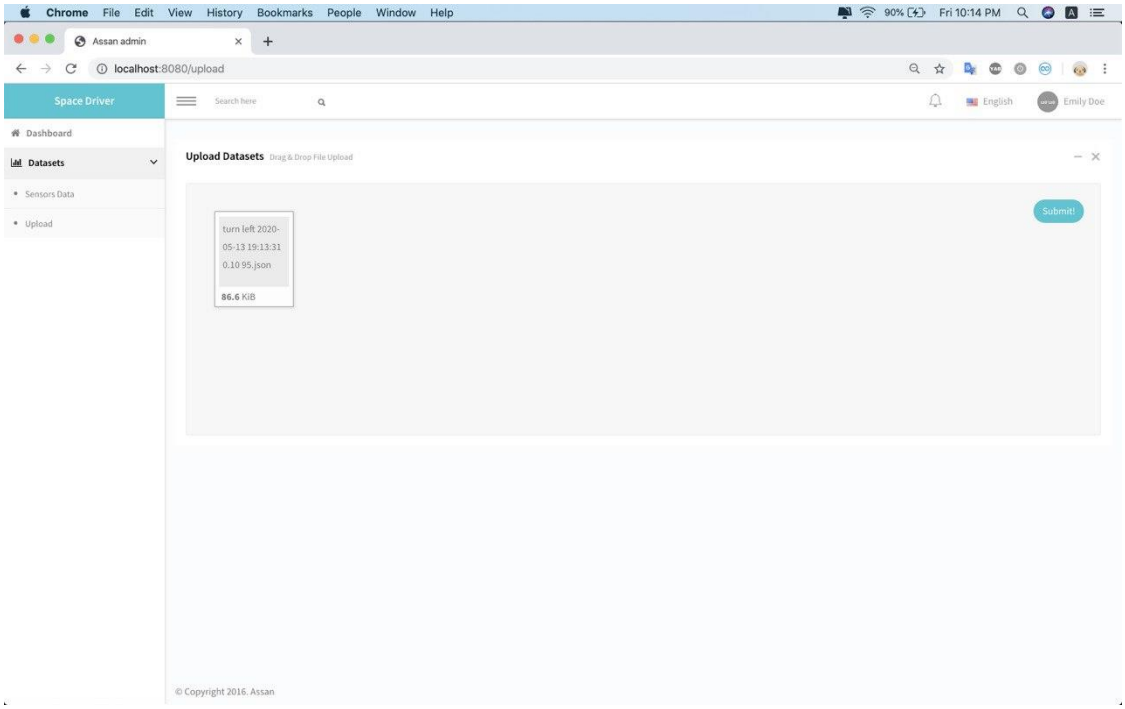


Рисунок 5.25 – Datasets (Upload Datasets).

На рисунках 5.26 – 5.28 наведено показники у вигляді графіків та діаграм, за конкретною таблицею-записом. Тут можна переглянути більш детально інформацію, отриману від сенсорів, на мобільних пристроях.

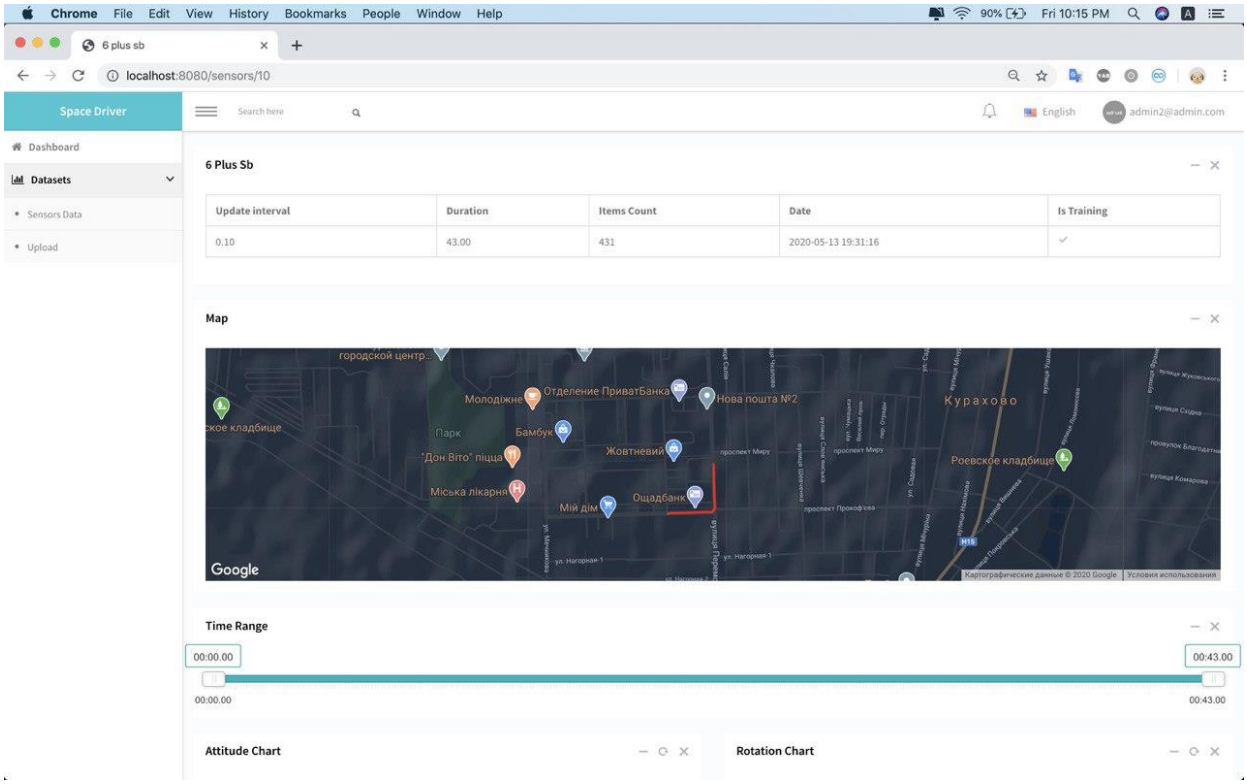


Рисунок 5.26 – Datasets (Upload Datasets).

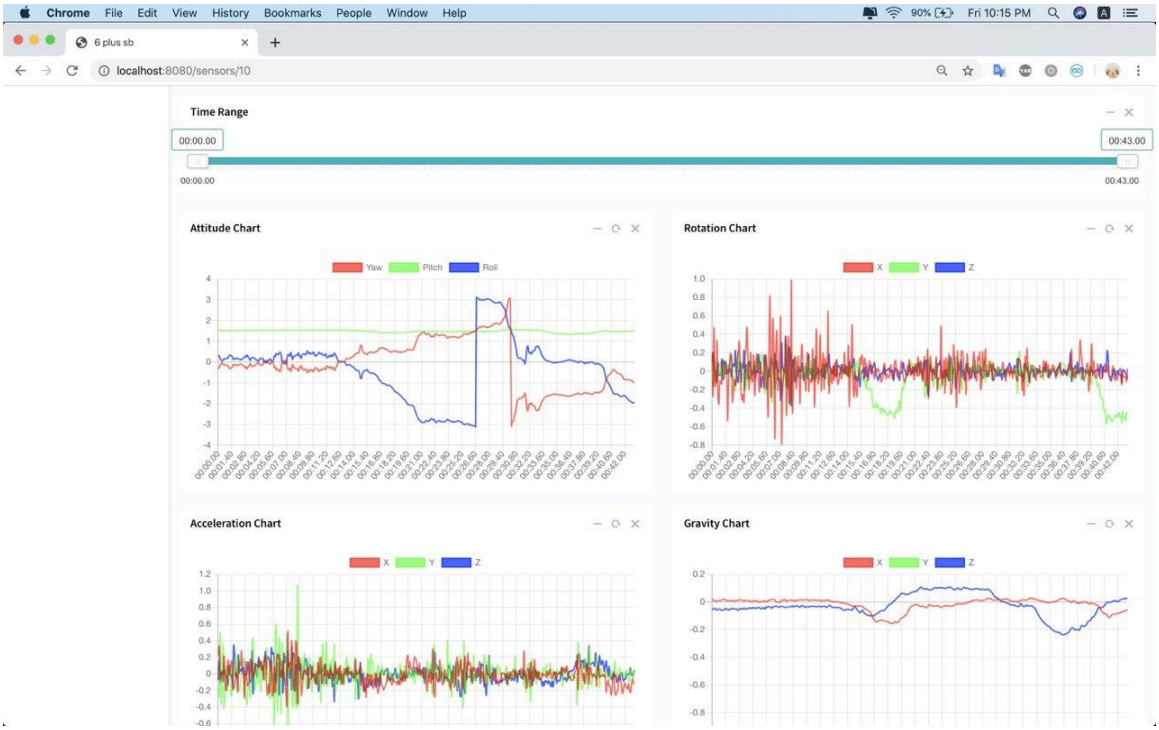


Рисунок 5.27 – Datasets (Upload Datasets).

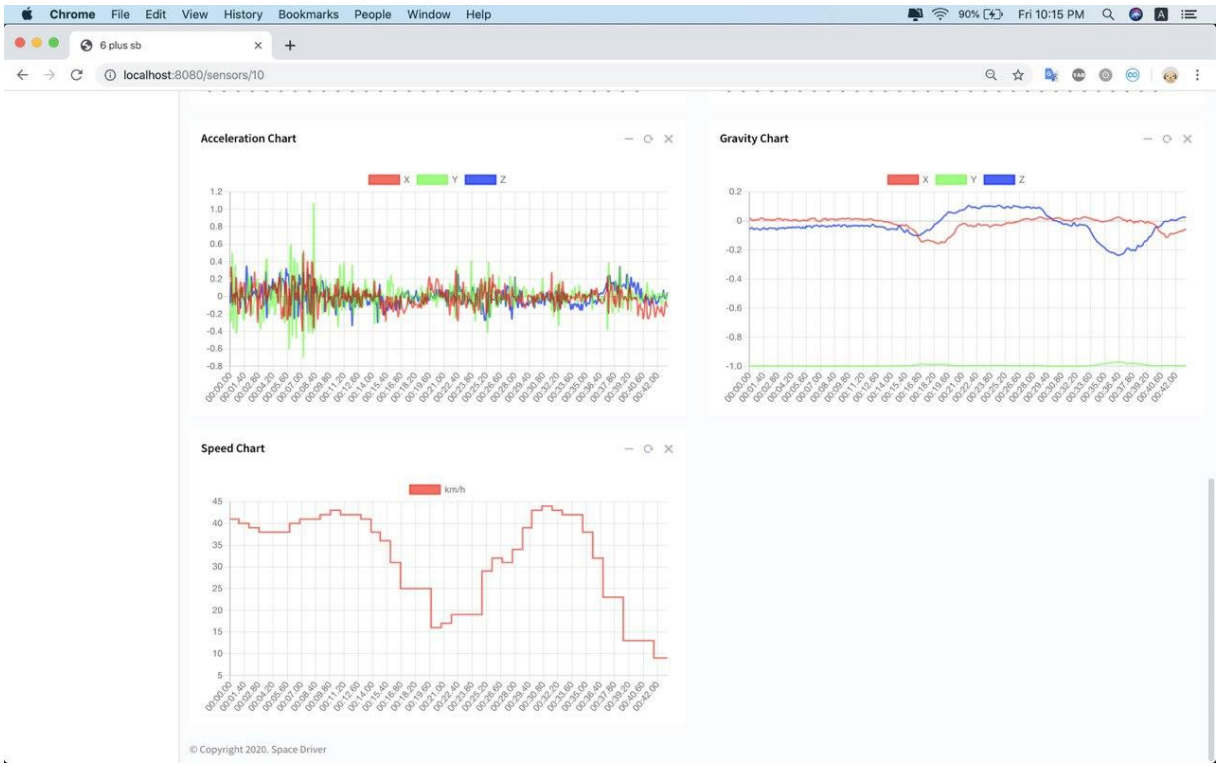


Рисунок 5.28 – Datasets (Upload Datasets).

# ВИСНОВКИ

У даній роботі було вивчено і проаналізовано проблему збору та аналізу даних у сфері транспортної телематики.

Було проведено огляд методів і засобів розробки програмної системи, а саме: мови програмування Swift, JavaScript, а також SQL. Для роботи було обрано середовище розробки xCode. Також використано СУБД MySQL та фреймворк Vapor.

Було виконано основну метою дипломної роботи – створено систему зчитування даних з сенсорів смартфону та аналізу інформації для подальшого її використання у сфері транспортної телематики.

Розроблено архітектуру програмного продукту, яка складається з бази даних, веб-додатку, та двох мобільних додатків. Один з мобільних додатків призначено для звичайних користувачів систем – водіїв, а інший призначений для адміністраторів системи, та має розширений функціонал.

Отже було виконано всі поставлені задачі, а саме:

1. Створено мобільний додаток клієнта (водія), який виступає у ролі навігатору, та надає доступ до інформації про стан доріг.

2. Створено мобільний додаток адміністратора, який базується на мобільному додатку клієнта, але з розширеними можливостями: відслідковування значень сенсорів у режимі реального часу, з їх візуальною інтерпретацією, та ручний режим запису даних з більшими можливостями.

3. Створено веб-додаток, який складається з форми авторизації (доступно тільки для адміністраторів), головного екрану на якому відображається стан системи, списку записаних даних сенсорів, екрану з детальною інформацією запису даних сенсорів, екран ручного завантаження даних сенсорів.

4. Створено API сервер, який зв'язує усі частини програмного забезпечення, має доступ до БД, керує системою авторизації, отримує дані, зчитані сенсорами, від мобільних додатків. Також виконує попередній та вихідний аналіз даних та їх

підготовку до наступного використання (фільтрація шумів, при завантаженні даних, їх реорієнтація і, як наслідок попередніх дій - аналіз стану доріг).

5. Створено базу даних, яка зберігає інформацію про користувачів системи, дані, отримані від сенсорів, дані аналізу стану доріг, а також дані, що потребуються для авторизації, історія дій в системі.

Проведено тестування програмного забезпечення в умовах міста та отримано ряд результатів, на основі яких побудовано мапу стану дороги. Програмна система успішно пройшла апробацію, а її результати подано у звіті.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ВЛАСОВ В. ТРАНСПОРТНАЯ ТЕЛЕМАТИКА В ДОРОЖНОЙ ОТРАСЛИ / В. ВЛАСОВ, В. БОГУМИЛ, Д. ЕФИМЕНКО,. – МОСКВА: МАДИ, 2013. – 80 с.

2. Labrador, M. Human Activity Recognition: Using Wearable Sensors and Smartphones / M. Labrador,, O. Lara Yejas., 2013. – 207 с.

3. Hou Z. Experimentation of 3D pavement imaging through stereovision. In Proceedings of the International Conference on Transportation Engineering / Z. Hou, K. Wang, W. Gong. – Chengdu, China, 2007. – 376 с.

4. Yagi K. Extensional smartphone probe for road bump detection / K. Yagi. – Busan, Korea, 2010.

5. Kim T. Review and analysis of pothole detection methods. / T. Kim, S. Ryu., 2014. – 603 с.

6. A road pavement monitoring system for anomaly detection using smart phones. In Big Data Analytics in the Social and Ubiquitous Context / F.Seraj, B. van der Zwaag, A. Dilo, T. Luarasi. – Berlin, Germany, 2014. – 128 с.

7. StarLine [Електронний ресурс]. – 1988. – Режим доступу до ресурсу: https://starline.in.ua/.

8. Пичаи С. MapsJavaScript API GotoConsole [Електронний ресурс] / СундарПичаи – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/javascript/infowindows.

9. Waze [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://www.waze.com/ru.

10. Усов В. Swift. Основыразработки приложений подiOS и macOS / ВасилийУсов., 2018. – 448 с.

11. Фленаган Д. JavaScript. Карманныйсправочник / Девід Фленаган., 2020. – 320 с.

12. xCode [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/xcode/.

13. UIKit [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/documentation/uikit.

14. Шварц Б. MySQL по максимуму. Оптимизация, репликация, резервноекопирование / БэронШварц., 2018. – 864 с.приложений / Ян Ф. Дарвин., 2019.

15. Vapor [Електронний ресурс] // 2016 – Режим доступу до ресурсу: https://vapor.codes/.

16. Google Maps Platform Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation.

# ДОДАТОК 1

Інструментальні засоби транспортної телематики

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61178_20Б 81-1

Аркушів 2

Київ – 2020

| Позначення | Найменування | Примітки |
|---|---|---|
| Документація | | |
| УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ТМ61178_20Б 81-1 | Записка.docx | Пояснювальна записка |
| Компоненти | | |
| УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ТМ61178_20Б 12-1 | Sources/Common/Services/Sensors/ | Модулі зчитуавння сенсорів смартфону |
| УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ТМ61178_20Б 12-2 | Sources/App/Controllers/Road Damage/RoadDamageService.swift | Модуль аналізу стану дороги |
| УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕ ПС_ТМ61178_20Б 13-1 | Опис.docx | Опис програми |

# ДОДАТОК 2

Інструментальні засоби транспортної телематики

Текст програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61178_20Б 12

Аркушів 11

Київ – 2020

```
protocol RecordPresenterInterface {

    // Recording flow
    var isRecord: Bool { get }

    func startRecord(with updateInterval: Double, isTraining: Bool)
    func stopRecord()
    func saveRecord(with name: String)
    func cancelRecord()
}

protocol RecordView: class {

    func update(stats: RecordStats)
    func resetStats()

    func showSaveRecordAlert()


//    func updateTimeLabel(value: Int64)

//    func showError(message: String)
}

final class RecordPresenter: RecordPresenterInterface {

    weak var view: RecordView!

    private let sensorsManager: SensorsManager
    private let recordsStorage: RecordsStorageInterface

    private var session: Session?
```

```
    private var recordStartDate: Date?

    public var isRecord: Bool {
        return session != nil
    }

    #if targetEnvironment(simulator)
    init(view: RecordView,
         sensorsManager: SensorsManager = SensorsManager(motionManager:
SimulatorMotionService(),
                                                         locationService:
LocationService.shared),
         recordsStorage: RecordsStorageInterface = RecordsStorage()) {

        self.view = view
        self.sensorsManager = sensorsManager
        self.recordsStorage = recordsStorage

        sensorsManager.delegate = self
    }
    #else
    init(view: RecordView,
         sensorsManager: SensorsManager = SensorsManager(),
         recordsStorage: RecordsStorageInterface = RecordsStorage()) {

        self.view = view
        self.sensorsManager = sensorsManager
        self.recordsStorage = recordsStorage

        sensorsManager.delegate = self
    }
    #endif

    func startRecord(with updateInterval: Double, isTraining: Bool) {
```

```
        session = Session(updateInterval: updateInterval, isTraining:
isTraining)
        sensorsManager.updateInterval = updateInterval

        sensorsManager.start()
    }

    func stopRecord() {
        sensorsManager.stop()

        view.resetStats()
        view.showSaveRecordAlert()
    }

    func saveRecord(with name: String) {
        defer {
            self.session = nil
        }

        guard let session = session else {
            Log.warning("Failed to save record, reason: session is nil")
            return
        }

        session.name = name

        recordsStorage.store(session: session)
    }

    func cancelRecord() {
        session = nil
    }

    // MARK: - Helpers
```

```
    private func makeRecordStats(motion: CMDeviceMotion?, location:
CLLocation?) -> RecordStats {

        let duration = session.map { Int64(-$0.datetime.timeIntervalSinceNow)
}
        let records = session?.items.count

        var attitude: String?
        var rotation: Int?
        var gravity: String?
        var acceleration: Int?

        if let motion = motion {
            attitude = String(format: "%.02f, %.02f, %.02f",
motion.attitude.roll, motion.attitude.pitch, motion.attitude.yaw)
            rotation = Int(abs(motion.rotationRate.x) +
abs(motion.rotationRate.y) + abs(motion.rotationRate.z))
            acceleration = Int(abs(motion.userAcceleration.x) +
abs(motion.userAcceleration.y) + abs(motion.userAcceleration.z))

            let isValidGravity = abs(motion.gravity.x) < 0.3 &&
abs(motion.gravity.z) < 0.5
            gravity = isValidGravity ? "Valid" : "Invalid"
        }

        var accuracy: Int?
        var speed: Int?

        if let location = location {
            accuracy = Int(location.horizontalAccuracy)
            speed = location.speed >= 0 ? location.speedKilometerPerHour :
nil
        }
```

```
        let isStable = motion != nil && location != nil
        let stability: RecordStats.Stability = isStable ? .good : .bad

        return RecordStats(duration: duration,
                           accuracy: accuracy,
                           attitude: attitude,
                           rotation: rotation,
                           gravity: gravity,
                           acceleration: acceleration,
                           speed: speed,
                           records: records,
                           activity: nil,
                           stability: stability)
    }
}

extension RecordPresenter: SensorsManagerDelegate {

    func sensorsManagerDidUpdateData(motion: CMDeviceMotion?, location:
CLLocation?) {
        let timestamp = sensorsManager.updateInterval *
Double(session?.items.count ?? 0)

        let sessionItem = SessionItem(deviceMotion: motion,
                                      location: location,
                                      timestamp: timestamp)

        session?.items.append(sessionItem)


        let recordStats = makeRecordStats(motion: motion, location: location)

        DispatchQueue.main.async {
```

```
            self.view.update(stats: recordStats)
        }
    }
}


import CoreMotion
import CoreLocation

protocol SensorsManagerDelegate: class {
    /// Executes on not main queue
    func sensorsManagerDidUpdateData(motion: CMDeviceMotion?, location:
CLLocation?)
}

final class SensorsManager: NSObject {

    // MARK: - Services

    private let motionManager: MotionServiceInterface
    private let locationService: LocationServiceInterface

    // MARK: - Configuration

    private lazy var underlyingQueue: OperationQueue = {
        let dispatchQueue = DispatchQueue(label: "com.medum.manager.sensors",
qos: .default)

        let queue = OperationQueue()
        queue.qualityOfService = .default
        queue.maxConcurrentOperationCount = 1
        queue.underlyingQueue = dispatchQueue

        return queue
```

```
    }()

    public var updateInterval: TimeInterval {
        get { motionManager.deviceMotionUpdateInterval }
        set { motionManager.deviceMotionUpdateInterval = newValue }
    }

    public weak var delegate: SensorsManagerDelegate?

    // MARK: - Init

    init(motionManager: MotionServiceInterface = CMMotionManager(),
         locationService: LocationServiceInterface = LocationService.shared)
{

        self.motionManager = motionManager
        self.locationService = locationService
        super.init()

        updateInterval = 0.2
    }

    public func requestPermission() {
        locationService.requestPermission()
    }

    public func start() {
        locationService.startUpdatingLocation()
        motionManager.startDeviceMotionUpdates(to: underlyingQueue,
                                               withHandler:
motionManager(didUpdateDeviceMotion:error:))
    }

    public func stop() {
```

```
        locationService.stopUpdatingLocation(isForce: false)
        motionManager.stopDeviceMotionUpdates()
    }

}

// MARK: - Updates

extension SensorsManager {

    func motionManager(didUpdateDeviceMotion motion: CMDeviceMotion?, error:
Error?) {
        delegate?.sensorsManagerDidUpdateData(motion: motion,
                                              location:
locationService.lastValidLocation)
    }
}


protocol LocationServiceObserver: class {

    func locationService(didUpdateLocation location: CLLocation)
    func locationServiceDidNotHavePermissions()
}

class LocationService: NSObject, LocationServiceInterface {

    public static let shared = LocationService()

    private let locationManager: CLLocationManager

    public var subscribers: [LocationServiceObserver] = []

    public var lastValidLocation: CLLocation?
```

```
    public var permission: CLAuthorizationStatus {
        return CLLocationManager.authorizationStatus()
    }

     override init() {
        locationManager = CLLocationManager()
        super.init()

        configure()
    }

    private func configure() {
        locationManager.desiredAccuracy =
kCLLocationAccuracyBestForNavigation
        locationManager.distanceFilter = 5

        locationManager.requestWhenInUseAuthorization()
        locationManager.allowsBackgroundLocationUpdates = true
        locationManager.pausesLocationUpdatesAutomatically = false
        locationManager.activityType = .automotiveNavigation

        locationManager.delegate = self
    }

    public func requestPermission() {
        locationManager.requestAlwaysAuthorization()
    }

    public func startUpdatingLocation() {
        if CLLocationManager.locationServicesEnabled() {
            locationManager.startUpdatingLocation()

        } else {
```

```
            subscribers.forEach { $0.locationServiceDidNotHavePermissions() }
        }
    }

    public func stopUpdatingLocation(isForce: Bool) {
        if isForce || subscribers.isEmpty {
            locationManager.stopUpdatingLocation()
        }
    }

    private func isValid(location: CLLocation) -> Bool {
        let age = -location.timestamp.timeIntervalSinceNow

        guard age <= 10 else {
            Log.info("Location: locaiton is old")
            return false
        }

        guard location.horizontalAccuracy > 0 else {
            Log.info("Location: latitidue and longitude values are invalid")
            return false
        }

        guard location.horizontalAccuracy < 70 else {
            Log.info("Location: accuracy is too low")
            return false
        }

        // Log.debug("Location quality is good enough")

        return true
    }
}
```

```
// MARK: - CLLocationManagerDelegate

extension LocationService: CLLocationManagerDelegate {

    func locationManager(_ manager: CLLocationManager, didUpdateLocations
locations: [CLLocation]) {

        guard let location = locations.last, isValid(location: location) else
{
            return
        }

        lastValidLocation = location

        subscribers.forEach { $0.locationService(didUpdateLocation: location)
}
    }

    func locationManager(_ manager: CLLocationManager, didChangeAuthorization
status: CLAuthorizationStatus) {
        switch status {
        case .authorizedWhenInUse:
            //You can resume logging by calling startUpdatingLocation here
            break

        default:
            break
        }
    }

    func locationManager(_ manager: CLLocationManager, didFailWithError
error: Error) {

        guard let locationError = error as? CLError else {
```

```
            Log.error("Unknown error, with type: \(type(of: error))")
            return
        }

        if locationError.code == .denied {
            //User denied your app access to location information.
            subscribers.forEach { $0.locationServiceDidNotHavePermissions() }
        }
    }

}

// MARK: - Obervers

extension LocationService {

    func subscribe(listener: LocationServiceObserver) {
        subscribers.append(listener)
    }

    func unsubscribe(listener: LocationServiceObserver) {
        subscribers.removeAll { $0 === listener }
    }

}
```



```
import Foundation

class RoadDamageSerivce {

    func predict(data: Session) -> [([SessionItem], Bool)] {
```

```
        var cursor = 0
        var result: [([SessionItem], Bool)] = []

        while true {

            let window = makeWindow(sessionItems: data.items, current:
cursor)
            cursor += window.0 + window.1
            if window.1 == 0 {
                return result
            }

            var minValue: Double = -99999
            var maxValue: Double =  99999

            let list = data.items[window.0..<window.0+window.1]
            for item in list {
                guard let acceleration = item.acceleration else {
                    break
                }

                guard let gravity = item.gravity else {
                    break
                }

                let resultAcceleration = acceleration.x * pow(gravity.x, 2) +
acceleration.y * pow(gravity.y, 2) + acceleration.z * pow(gravity.z, 2)

                if resultAcceleration < minValue {
                    minValue = resultAcceleration
                }

                if resultAcceleration > maxValue {
                    maxValue = resultAcceleration
```

```
                }
            }

            let amplitude = maxValue - minValue
            let greatestVal = max(abs(maxValue), abs(minValue))

            let sumSpeed = list.compactMap { $0.location?.speed }
            let speed = sumSpeed.reduce(0, +) / Double(sumSpeed.count)

            var isAnomaly = false

            if speed < 30 {
                if greatestVal > 0.2 {
                    isAnomaly = true
                } else {
                    isAnomaly = false
                }

            } else {
                if greatestVal > 0.42 {
                    isAnomaly = true
                } else {
                    isAnomaly = false
                }
            }

            result.append((Array(list), isAnomaly))
        }
    }

    func makeWindow(sessionItems: [SessionItem], current: Int) -> (Int, Int)
{

        var index = current
```

```
        var counter = 0
        while index < sessionItems.count && counter < 10 {

            guard let speed = sessionItems[index].location?.speed, speed > 10
else {
                index += 1
                continue
            }

            counter += 1
        }

        if index == sessionItems.count || counter < 10 {
            return (index - counter, counter)
        }

        var sum = 0.0
        var handledItems = 0.0
        let average = Array(sessionItems[index..<index+counter].map { item ->
Double in
            let speed = item.location?.speed ?? 0
            sum += speed
            handledItems += 1

            return sum / handledItems
        }.suffix(7))

        let predicts: [(Double, Int)] = [(20, 10), (25, 9), (30, 8), (50, 7),
(60, 6), (68, 5), (74, 4), (80, 3)]

        var result: (Double, Int) = (.greatestFiniteMagnitude, .max)
        for i in 0 ..< average.count {
```

```
            var delta = predicts.map { (abs($0.0 - average[i]), abs($0.1 -
i)) }
            var local: (Double, Int) = (.greatestFiniteMagnitude, .max)
            for j in 0 ..< delta.count where  local.0 > delta[j].0 *
Double(delta[j].0) {
                local = (delta[j].0 * Double(delta[j].0), j)
            }

            if result.0 > local.0 {
                result = local
            }
        }

        return (index, predicts[result.1].1)
    }
}
```

# ДОДАТОК 3

Інструментальні засоби транспортної телематики

Опис програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61178_20Б 13-1

Аркушів 8

Київ – 2020

# АНОТАЦІЯ

Додаток надає можливість зчитувати та аналізувати показники сенсорів смартфона, та використовує отримані дані для встановлення аномалій на дорозі.

Розроблене програмне забезпечення виступає у ролі сучасного мобільного дорожнього навігатора, з можливістю перегляду стану дороги та перегляду стану дороги на прокладеному маршруту.

Програмний продукт створений мовою програмування Swift з використанням фреймворків UIKit, CoreLocation, CoreMotion, Vapor. Розробка інтерфейсу додатку відбувалася за допомогою Storyboard, HTML\CSS. В якості середовища розробки – xCode.

# ЗМІСТ

-4-

# ЗАГАЛЬНІ ВІДОМОСТІ

Розроблений продукт є мобільним дорожнім навігатором, що має додаткові можливості, наприклад, перегляд стану дороги, зчитування показників з сенсорів смартфона. Також для зручного аналізу отриманих даних було розроблено веб-додаток, якій містить візуалізацію показників з сенсорів.

Користувач для роботи з додаток повинен мати мобільний телефон на базі iOS.

Для використання додатку потрібно встановити розроблений додаток на смартфон, та ноутбук або комп'ютер. Для встановлення та використання додатку не потрібно ніяких втручань, що потребували би знань програмування.

-5-

# ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб покликаний вирішити задачу збір та аналіз великої кількості даних на дорозі, для подальшої оптимізації процесів на дорозі. Це було реалізовано за допомогою кількох функцій системи, а саме:

— моделювання сигналу прискорення пристрою за допомогою сенсорів смартфона;

— відображення графічної візуалізації отриманих даних;

— пошук аномалій на дорозі

-6-

# ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний продукт складається з клієнтської та серверної частини.

Загальний принцип роботи додатку такий:

1) користувач відкриває карту,
2) місце положення та стан дороги підсвічується на карті;
3) користувач нажимає на кнопку подорожі;
4) натиснути на місце призначення;
5) прокладений маршрут підсвічується на карті;
6) підтверджує прокладений маршрут;

В першу чергу завантажується карта, після чого додаток показує місце положення користувача та стан дороги поблизу. Перед початком подорожі користувач натискає відповідну кнопку в нижньому лівому куту екрана. Після цього додаток переходить у стан подорожі, тепер додаток автоматично скролить за позицією користувача, а на місці кнопки подорожі з'являється спідометр. Також пыд час подорожі, додаток парелельно зчитує показники з сенсорів смартфона та відправляє на сервер, для аналізу стану дороги та трафіка. Якщо користувачу потрібно прокласти маршрут, він має натиснути на відповідне місце призначення на карті. Після чого додаток прокладе маршрут на місця призначення та виведе інформацію про даний маршрут, а саме: приблизинй час подорожі, дистанція до місця призначення, та стан доріг на цьому маршруті. Користувач натискає кнопку старт, після чого додаток переходить у стан подорожі, якщо він ще не був у цьому стані. По завершенню маршрута користувач має натиснути на маркер місця призначення, після чого маршрут пропаде з карти.

-7-

# ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання програмного продукту потрібно мати сматфон на операційній системі iOS, та ком'ютер на операційній системі macOS.

Кінцевим користувачем системи потрібен лише смартфон на операційній системі iOS, та не потрібно ніяких додаткових програмних засобів для використання системи.

-8-

# ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— показники сенсорів;

— карта Google Map

Вихідними даними є:

— схеми

— діаграми

— інформація про стан доріг

— інформація про завантаженість дороги

— маршрут до місця призначення з інформацією про цей маршрут